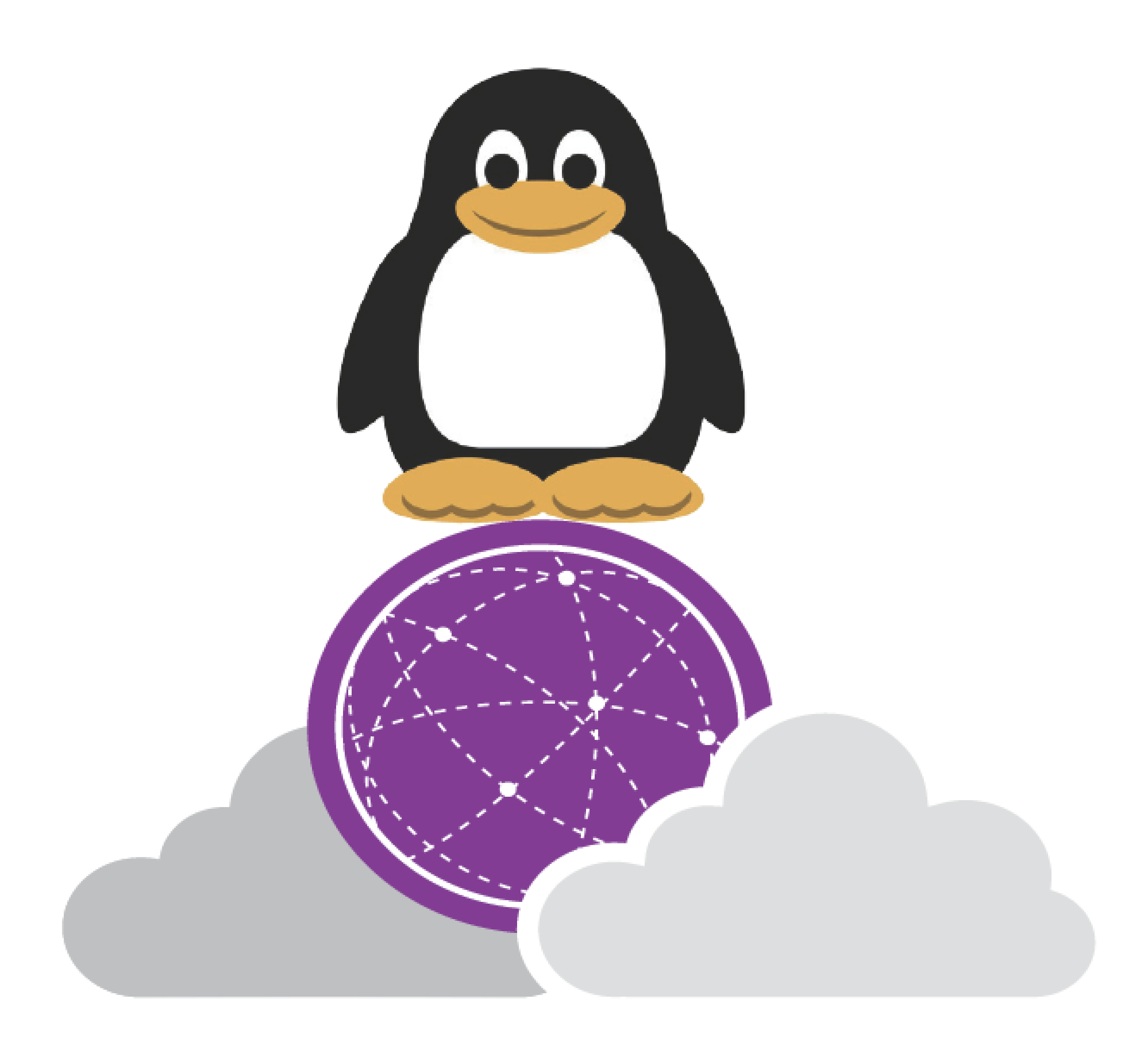
知名Linux运维专家**吴光科**力作

提供丰富实例，可直接应用于生产环境；提供配套源代码，可直接动手实践，二次开发





Exposure Linux Enterprise Operation and Maintenance



Linux企业运维实战

吴光科◎编著

Wu Guangke

丁超
百度集团公司技术经理

李志明
京东商城高级运维经理

王帅
阿里巴巴资深运维架构师

蔡正雄
京峰教育首席运营官

华宇飞
中国教师研修网高级运维总监

贾云龙
乐博学院首席执行官

萧田国
高效运维社区&DevOpsDays中国联合发起人

联袂推荐

清華大學出版社

清华开发者书库

曝光：Linux企业运维实战

吴光科　编著

清华大学出版社
北　京

内 容 简 介

本书系统地论述了 Linux 运维领域的各种技术，主要内容包括最新版 Linux 系统安装的完整过程、系统启动原理、系统必备的命令、系统管理、初学者必备软件管理、基础服务实战、生产环境 LAMP、Redis 实战及备份、性能优化、Zabbix 企业级分布式监控、Nginx 高性能 Web 服务器实战及 Nginx 相关 location、rewrite 规则、日志分析实战、Nginx 性能调优、自动化运维技术实战、shell 编程入门、shell 编程企业实战案例剖析、shell 编程企业案例详解、Puppet、Ansible 案例深入剖析、企业高性能负载均衡技术 LVS、keepalived 高可用集群满足千万 PV 门户网站架构、Haproxy 高性能负载均衡、构建企业级自动化部署平台 Jenkins、CI/CD 自动部署及交付、Docker 虚拟化企业实战、Docker 镜像、容器、DokerFile、Docker 生产环境一键脚本配置、Docker Pipework 网络实战、Openstack 私有云、Openstack 企业实战、构建企业 Openstack 私有云、Openstack 常见问题排错等核心内容。

本书适合作为系统管理员、网络管理员、Linux 运维工程师及网站开发、测试、设计人员等的参考用书。

图书在版编目(CIP)数据

曝光：Linux 企业运维实战/吴光科编著．—北京：清华大学出版社，2018
(清华开发者书库)
ISBN 978-7-302-48484-4

Ⅰ．①曝…　Ⅱ．①吴…　Ⅲ．①Linux 操作系统　Ⅳ．①TP316.85

中国版本图书馆 CIP 数据核字(2017)第 230262 号

责任编辑： 盛东亮
封面设计： 李召霞
责任校对： 李建庄
责任印制： 李红英

出版发行： 清华大学出版社
网　　址： http://www.tup.com.cn，http://www.wqbook.com
地　　址： 北京清华大学学研大厦 A 座　　**邮　　编：** 100084
社 总 机： 010-62770175　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载： http://www.tup.com.cn，010-62795954
印 装 者： 三河市铭诚印务有限公司
经　　销： 全国新华书店
开　　本： 186mm×240mm　　**印　　张：** 36.25　　**字　　数：** 812 千字
版　　次： 2018 年 5 月第 1 版　　**印　　次：** 2018 年 5 月第 1 次印刷
印　　数： 1～2000
定　　价： 99.00 元

产品编号：076280-01

前 言

PREFACE

为什么要写这本书

为什么写《曝光：Linux 企业运维实战》这本书？这要从我的经历说起。我出生在贵州省一个贫困的小山村，从小经历了山里砍柴、放牛、挑水、做饭、日出而作、日落而归的朴素生活，看到父母一辈子都在小山村里，没有见过大城市，所以从小立志要走出大山，要让父母过上幸福的生活！

正是这样一个信念让我不断地努力，大学毕业至今，在“北漂”的 IT 运维路上走过了 9 年多，从最初小公司的网管到国企机关、图吧、研修网、京东商城等一线 IT 企业，分别担任过 Linux 运维工程师、Linux 运维架构师、运维经理，到今天创办了京峰教育培训机构。

这一路走来，要感谢生命中遇到的每一个人，是大家的帮助，让我不断地进步和成长，也让我明白了一个人活着不应该只为自己和自己的家人，而是要为这个社会，哪怕只能对社会贡献一点点的价值，人生就是精彩的。为了帮助更多的人通过技术改变自己的命运，我决定编写《曝光：Linux 企业运维实战》这本书。虽然市面上有很多关于 Linux 的书籍，但是很难找到一本关于 Linux 企业生产环境、企业自动化运维、云计算、虚拟化等主流技术的书籍，这是我编写本书的初衷！

本书读者对象

系统管理员、网络管理员、在校大学生、Linux 运维工程师、Linux 系统管理人员及从事云计算、网站开发、测试、设计的人员。

如何阅读本书

全书分为三篇，第一篇：Linux 基础篇，包括第 1～8 章，主要内容为 Linux 快速入门、Linux 发展及系统安装、CentOS 系统管理、Linux 必备命令、Linux 用户及权限管理、Linux 软件包企业实战、Linux 磁盘管理、Linux 文件服务器企业实战。俗话说“看百遍不如跟着书操作一遍”，所以笔者建议读者在阅读本书时，应根据本书中的提示和各种操作案例，使用真实服务器或者虚拟机实战练习，这样可以更好地理解每条命令及涉及的各个步骤，从而更加高效地学习，把基础打牢。

第二篇：Linux 进阶篇，包括第 9～14 章，主要内容为 HTTP 协议详解、Apache Web 服务器企业实战、MySQL 服务器企业实战、LAMP 企业架构实战、Zabbix 分布式监控企业实战、Nginx Web 服务器企业实战。

第三篇：Linux 高级篇，包括第 15～25 章，主要内容为 Linux 性能优化企业实战、大数据量备份企业实战、shell 企业编程基础、shell 编程高级企业实战、自动化运维发展前景、Puppet 自动运维企业实战、Ansible 自动运维企业实战、Jenkins 持续集成企业实战、Linux 高可用集群实战、实战 Docker 虚拟化技术、Openstack＋KVM 构建企业私有云。

勘误和支持

尽管笔者花费了大量的时间和精力来核对书中的各个代码和语法，但其中难免还会存在一些纰漏，恳请读者指正和批评。如果大家发现有任何问题，都请及时反馈给我，相关信息可以发到个人邮箱 wgkgood@163.com，也可以加入本书支持 QQ 群（432241666、418600627），我会竭尽全力为读者服务。

致谢

感谢 Linux 之父——Linus Torvalds，Linus Torvalds 不仅创造了 Linux 系统，而且影响了整个开源世界，同时也影响了我的一生！

感谢我亲爱的父母，含辛茹苦地把我们兄弟三人抚养长大，是他们对我无微不至地照顾，让我有更多的精力和动力去工作，去帮助更多的人！

感谢挚友潘彦伊、周飞、何红敏、周孝坤、杨政平、王帅、李强、刘继刚、常青帅、孙娜、吴俊、李芬伦、陈洪刚、黄宗兴、代敏、杨永琴、姚钗及其他挚友们这么多年来对我的信任和支持，从始至终一直都在默默地支持我。

感谢清华大学出版社盛东亮编辑及各位工作人员，在他们的信任、支持和帮助下，我才能如此顺利地完成全部书稿。

感谢腾讯公司腾讯课堂周唯经理及平台所有的老师，感谢乐博学院 CEO 贾云龙及乐博学院的各位老师，感谢 51CTO 学院院长一休及全体工作人员对我及京峰教育培训机构的大力支持！

感谢京峰教育培训机构的每位学员对我的支持和鼓励，希望他们都学有所成，最终成为社会的中流砥柱！感谢京峰教育培训机构 COO 蔡正雄！感谢京峰教育培训机构的陈老师、张老师、华华老师、品茶老师、灿哥、陶老师、胡老师及全体老师和助教、班长、副班长，是他们的大力支持，让京峰教育能够帮助更多的小伙伴！

最后要感谢我的爱人黄小红，是她一直在背后默默地支持我、鼓励我，让我有更多的精力和时间去完成这本书，有她真好！

吴光科

2018 年 2 月

目 录
CONTENTS

第一篇 Linux 基础篇

第一篇　Linux 基础篇

Linux 基础篇总共包含 8 个章节，第 1～8 章学习内容分别为 Linux 快速入门、Linux 发展及系统安装、CentOS 系统管理、Linux 必备命令、Linux 用户及权限管理、Linux 软件包企业实战、Linux 磁盘管理、Linux 文件服务器企业实战。

读者通过对基础篇 8 个章节的深入学习，可以了解 Linux 发展历程，了解 Linux 发行版之间的特性以及 Linux 内核命名规范，基于虚拟机环境手动安装的 CentOS 操作系统，能够快速上手，快速地入门 Linux。

同时读者能够熟练掌握 Linux 操作系统完整的启动流程，掌握 Linux 操作系统用户和组管理的机制，通过对 Linux 系统文件及目录进行权限定制和分配，从而提升 Linux 操作系统的使用安全，更加保证系统的稳定性。

对 Linux 必备命令的掌握程度，直接决定后期对 Linux 系统能否进行娴熟的操作，同时掌握 Linux 高效学习的大绝招，养成学习 Linux 的习惯和方法，对后期的 Linux 学习能起到事半功倍的效果。

俗话说"基础不牢，地动山摇"，熟练掌握 Linux 基础篇的相关内容，能够独立维护和管理企业 Linux 操作系统，为后期维护企业生产环境服务器打下坚实的基础。

第 1 章 Linux 快速入门



Linux 是一套免费使用和自由传播的类 UNIX 操作系统，是一个基于 POSIX 移植操作系统接口（portable operating system interface of UNIX，POSIX）和 UNIX 的多用户、多任务、支持多线程和多 CPU 的操作系统。

目前被广泛应用于企业服务器、Web 网站平台、大数据、虚拟化、Android、超级计算机等领域，未来 Linux 将应用于各行各业，如云计算、物联网、人工智能等。

本章将向读者介绍 Linux 发展简介、Linux 发行版特点、32 位及 64 位 CPU 特性以及 Linux 内核命名规则等内容。

1.1 为什么要学习 Linux

我们为什么要学习 Linux？我们目前的处境是什么？我们想达到什么样的目标？在谈到这三个问题时，相信每个人都有自己的答案，我们来自不同的家庭，各自经历也都不一样，但最终的目标都是希望通过学习技术，提升自己的专业技能，真正做一个对社会有贡献的人。

想想我们从刚步入学堂的那一刻起，心里就狠狠下定决心，以后不管做什么，都要有一番出息。可是二三十年过去了，我们收获了什么，得到了什么，到底是在追求什么？方向又在哪里呢？

在生活中各种挫折、感情以及很多零碎的事情，让我们很难静下心来学习，当我们某天突然惊醒，年少已不在。所以今天就下定决心，现在就要学习，去行动，去改变。

人生最可怕的是在自以为舒适的地方待得太久，等到外界改变来的时候，已经晚了，我们不能像温水煮青蛙一样，待在温水里，觉察不到周围事物的变化，最终被社会淘汰，如图 1-1 所示。



图 1-1 温水煮青蛙

1.2　Linux 操作系统简介

Linux 操作系统是基于 UNIX 以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统，Linux 能运行各种工具软件、应用程序以及网络协议，它支持安装在 32 位和 64 位 CPU 硬件上。

通常来讲，Linux 这个词本身只表示 Linux 内核，但是人们已经习惯用 Linux 来形容整个基于 Linux 内核的操作系统，并且是一种使用 GNU 通用公共许可证（GNU general public license，GPL）工程各种工具和数据库的操作系统。

GNU 是 GNU is not UNIX 的缩写，UNIX 是一种广泛应用的商业操作系统，由于 GNU 将要实现以 UNIX 系统的接口标准，因此 GNU 计划可以分别开发不同的操作系统部件，并且采用了部分当时已经可自由使用的软件。

为了保证 GNU 软件可以自由地使用、复制、修改和发布，所有的 GNU 软件都在一份禁止其他人添加任何限制的情况下授权所有权利给任何人的协议条款里，我们把这个条款称为 GPL。

1991 年的 10 月 5 日，Linux 创始人 Linus Torvalds 在 comp. os. minix 新闻组上发布消息，正式向外宣布 Linux 内核的诞生，1994 年 3 月 Linux 1.0 发布，代码量 17 万行，当时是完全按照自由免费的协议发布，随后正式采用 GPL 协议，目前 GPL 协议版本包括 GPLv1、GPLv2、GPLv3 以及未来的 GPLv4、GPLv5 等。

1.3　Linux 操作系统优点

随着 IT 产业的不断发展，Linux 操作系统应用领域越来越广泛，尤其是近年来 Linux 在服务器领域飞速地发展，这主要得益于 Linux 操作系统具备以下优点：

- 开源免费；
- 系统迭代更新；
- 系统性能稳定；
- 安全性高；
- 多任务，多用户；
- 耗资源少；
- 内核小；
- 应用领域广泛；
- 使用及入门容易。

1.4 Linux 操作系统发行版

学习 Linux 操作系统，需要选择不同的发行版本，Linux 操作系统发行版本众多，目前 Linux 操作系统主流发行版本包括 Red Hat Linux、CentOS、Ubuntu、SUSE Linux、Fedora Linux 等，具体发行版本之间的区别如下。

1. Red Hat Linux

Red Hat Linux 是最早的 Linux 发行版本之一，同时也是最著名的 Linux 版本，Red Hat Linux 已经创造了自己的品牌，也就是读者经常听到的“红帽操作系统”。Red Hat 于 1994 年创立，目前该公司在全世界有 3000 多人，一直致力于研发开放的源代码体系，向用户提供一套完整的服务，这使得它特别适合在公共网络中使用。这个版本的 Linux 也使用最新的内核，同时还拥有大多数人都需要使用的主体软件包。

Red Hat Linux 发行版操作系统的安装过程非常简单，图形安装过程提供简易设置服务器的全部信息，磁盘分区过程可以自动完成，还可以通过图形界面(graphical user interface，GUI)完成安装，即使对于 Linux 新手来说这些过程都非常简单。后期如果想批量安装 Red Hat Linux 系统，可以通过批量的工具来实现快速安装。

2. CentOS

社区企业版操作系统(community enterprise operating system，CentOS)是 Linux 发行版之一，它是来自于 Red Hat Enterprise Linux 依照开放源代码编译而成的。由于出自同样的源代码，因此有些要求高度稳定性的服务器以 CentOS 替代商业版的 Red Hat Enterprise Linux 使用。

CentOS 与 Red Hat Linux 不同之处在于 CentOS 并不包含封闭的源代码软件，可以开源免费使用，这使得它得到运维人员、企业、程序员的青睐。CentOS 发行版操作系统是目前企业使用最多的服务器领域的系统之一，2016 年 12 月 12 日，基于 Red Hat Enterprise Linux 的 CentOS Linux 7 (1611)系统正式对外发布。

3. Ubuntu

Ubuntu 是一个以桌面应用为主的 Linux 操作系统，其名称来自非洲南部祖鲁语或豪萨语的“ubuntu”一词(译为吾帮托或乌班图)，意思是“人性”“我的存在是因为大家的存在”，代表非洲传统的一种价值观。

Ubuntu 基于 Debian 发行版和 GNOME 桌面环境，Ubuntu 发行版操作系统的目标在于为用户提供一个最新的、同时稳定地以开放自由软件构建而成的操作系统，目前 Ubuntu 具有庞大的社区力量，用户可以很方便地从社区获得帮助。

4. SUSE Linux

SUSE(发音//suːsə/)是指 SUSE Linux，是德国 SuSE Linux AG 公司发行维护的 Linux 发行版，是属于此公司的注册商标。2003 年 11 月 4 日，Novell 公司表示将会对

SUSE 提出收购，收购的工作于 2004 年 1 月已完成。

Novell 公司也向大家保证 SUSE 的开发工作仍会继续下去，Novell 更把公司内全线电脑的系统换成 SUSE Linux，并同时表示将会把 SUSE 特有而优秀的系统管理程序- YaST2 以 GPL 授权释出。

5. Fedora Linux

Fedora 是一个知名的 Linux 发行版，是一款由全球社区爱好者构建的、面向日常应用的、快速、稳定、强大的操作系统。它允许任何人自由地使用、修改和重发布，无论现在还是将来。它由一个强大的社群开发，这个社群的成员以自己的不懈努力，提供并维护自由、开放源码的软件和开放的标准。

Fedora 约每 6 个月会发布新版本，美国当地时间 2015 年 11 月 3 日，北京时间 2015 年 11 月 4 日，Fedora Project 宣布 Fedora 23 正式对外发布，2017 年 6 月发布了 Fedora 26 版本。

1.5 32 位与 64 位操作系统的区别

学习 Linux 操作系统之前，需要掌握计算机基础知识。计算机内部对数据的传输和储存都是使用二进制，二进制是计算技术中广泛采用的一种数制，而 b(位)则表示二进制位，二进制数是用 0 和 1 两个数码来表示的数。基数为 2，进位规则是“逢二进一”，0 或 1 分别表示 1 位二进制数。

b(位)是计算机最小单位，而 B(字节)是计算机中数据处理的基本单位，转换关系为 1B=8b，4B=32b。

随着计算机技术的发展，尤其是中央处理器(central processing unit，CPU)技术的变革，CPU 的位数指的是通用寄存器(general-purpose registers，GPRs)的数据宽度，也就是处理器一次可以处理的数据量多少。

主流 CPU 处理器分为 32 位 CPU 处理器和 64 位 CPU 处理器，32 位 CPU 处理器可以一次性处理 4 个字节的数据量。而 64 位处理器一次性处理 8 个字节的数据量(1B=8b)，64 位 CPU 处理器对计算机处理器在 RAM(随机存取储存器)里处理信息的效率比在 32 位 CPU 里做了很多优化，效率更高。

x86_32 位操作系统和 x86_64 操作系统的区别也是基于 CPU 位数的支持，具体区别如下：

- 32 位操作系统表示 32 位 CPU 对内存寻址的能力；
- 64 位操作系统表示 64 位 CPU 对内存寻址的能力；
- 32 位的操作系统安装在 32 位 CPU 处理器和 64 位 CPU 处理器上；
- 64 位操作系统只能安装在 64 位 CPU 处理器上；
- 32 位操作系统对内存寻址不能超过 4GB；
- 64 位操作系统对内存寻址可以超过 4GB，企业服务器更多安装 64 位操作系统，支持更多内存资源的利用；

- 64 位操作系统是为高性能处理需求设计，满足数据处理、图片处理、实时计算等领域需求；
- 32 位操作系统是为普通用户设计，满足普通办公、上网冲浪等需求。

1.6　Linux 内核命名规则

Linux 内核是 Linux 操作系统的核心，一个完整的 Linux 发行版包括进程管理、内存管理、文件系统、系统管理、网络管理等部分。

Linux 内核官网可以下载 Linux 内核版本、现行版本、历史版本，可以了解版本与版本之间的特性。

Linux 内核版本命名在不同的时期有不同的命名规范，其中在 2.X 版本中，X 如果为奇数表示开发版、X 如果为偶数表示稳定版，从 2.6.X 以及 3.X，内核版本命名就没有严格的约定规范。

从 Linux 内核 1994 年发布 1.0 版本到目前主流的 2.6、3.X 版本，4.X 属于开发调试阶段，查看 Linux 操作系统内核如图 1-2 所示。

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# uname -a
Linux www-jfedu-net 2.6.32-642.15.1.el6.x86_64 #1 SMP Fri Feb
4 GNU/Linux
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# cat /proc/version
Linux version 2.6.32-642.15.1.el6.x86_64 (mockbuild@c1bm.rdu2
(Red Hat 4.4.7-17) (GCC) ) #1 SMP Fri Feb 24 14:31:22 UTC 20
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
```

图 1-2　操作系统内核

Linux 内核命名格式为“R.X.Y-Z”，其中 R、X、Y、Z 命名含义如下：

- 数字 R 表示内核版本号，版本号只有在代码和内核有重大改变的时候才会改变，到目前为止有 4 个大版本更新。
- 数字 X 表示内核主版本号，主版本号根据传统的奇偶系统版本编号来分配，奇数为开发版，偶数为稳定版。
- 数字 Y 表示内核次版本号，次版本号是无论在内核增加安全补丁、修复 bug、实现新的特性或者驱动时都会改变。
- 数字 Z 表示内核小版本号，小版本号会随着内核功能的修改、bug 修复而发生变化。

官网内核版本如图 1-3 所示，mainline 表示主线开发版本，stable 表示稳定版本，稳定版本主要由 mainline 测试通过而发布。longterm 表示长期支持版，会持续更新及 bug 修复，如果长期版本被标记为 EOL(end of life)，则表示不再提供更新。

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel: 4.10.12

mainline:	**4.11-rc7**	2017-04-16	[tar.xz]	[pgp]	[patch]		[view diff]	[browse]	
stable:	**4.10.12**	2017-04-21	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	**4.9.24**	2017-04-21	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	**4.4.63**	2017-04-21	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	**4.1.39**	2017-03-13	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	**3.18.49 [EOL]**	2017-04-18	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	**3.16.43**	2017-04-04	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	**3.12.73**	2017-04-13	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	**3.10.105**	2017-02-10	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]

图 1-3　官网内核版本

第 2 章



Linux 发展及系统安装

互联网飞速发展，用户对网站体验的要求也越来越高，目前主流 Web 网站后端承载系统均为 Linux 操作系统，Android 手机也是基于 Linux 内核而研发的，企业大数据、云存储、虚拟化等先进技术也均是以 Linux 操作系统为载体，以满足企业的高速发展。

本章向读者介绍 Linux 发展前景、Windows 与 Linux 操作系统区别、硬盘分区介绍、CentOS 7 Linux 操作系统安装图解以及菜鸟学好 Linux 必备大绝招等内容。

2.1 Linux 发展前景及就业形势

据权威部门统计，未来几年内我国软件行业的从业机会十分庞大，中国每年对 IT 软件人才的需求将达到 200 万人左右。而 Linux 专业人才的就业前景更是广阔，据悉在未来 5～10 年内 Linux 专业人才的需求将达到 150 万人，尤其是对于有 Linux 行业经验的人才，资深的 Linux 工程师非常缺乏，薪资也非常诱人，平均月薪 15000～25000 元，甚至更高，Linux 行业薪资如图 2-1 所示。

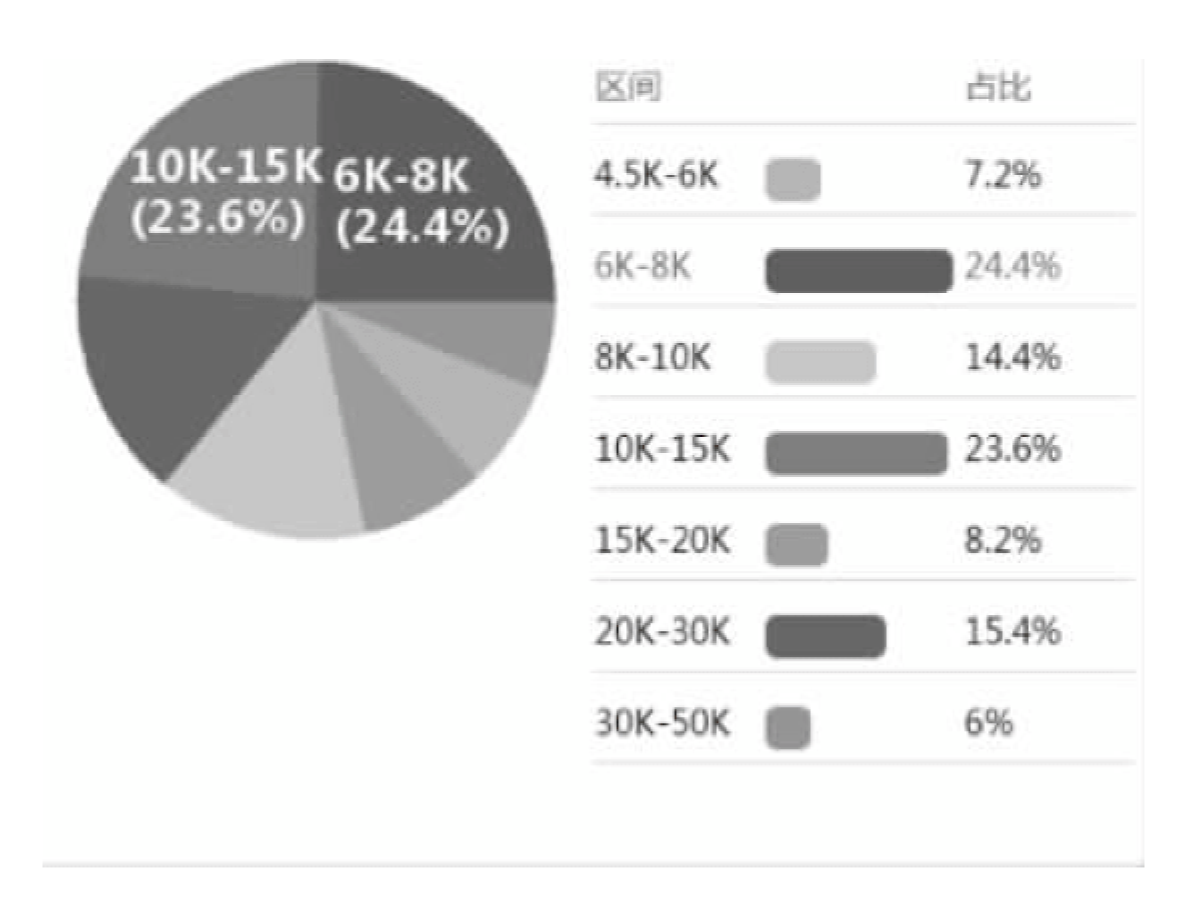


图 2-1　Linux 行业薪资

2.2 Windows 操作系统简介

为什么要学习 Windows 操作系统？了解 Windows 系统结构，可以让读者快速学习 Linux 操作系统，通过对比学习的方法，可以更快地学会 Linux。

计算机硬件组成包括 CPU、内存、网卡、硬盘、DVD 光驱、电源、主板、显示器、鼠标等设备，计算机硬件是不能直接被用户使用的，需要安装操作系统和驱动程序，才可进行操作、办公、上网冲浪等。

驱动程序主要指的是设备驱动程序（device driver），是一种可以使计算机系统和设备通信的特殊程序，相当于硬件的接口，操作系统只有通过这个接口，才能控制硬件设备，进行资源调度。

Windows 操作系统主要以窗口形式对用户展示，操作系统须安装在硬盘上，安装系统之前需对硬盘进行分区并格式化，默认 Windows 操作系统安装在 C 盘分区，D 盘分区用于存放数据文件。

通俗地讲，安装操作系统时，需要对磁盘进行格式化，格式化需要指定格式化的类型，告诉操作系统如何去管理磁盘空间，文件如何存放，如何查找及调度。操作系统不知道怎么存放文件以及文件结构，因此文件系统的概念就诞生了。

文件系统是操作系统用于明确磁盘或分区上文件的方法和数据存储结构，文件系统由 3 部分组成：与文件管理有关软件、被管理文件以及实施文件管理所需数据结构。

Windows 操作系统的文件系统类型一般有 FAT、FAT16、FAT32、NTFS 等，不同的文件系统类型有不同的特性。例如 NTFS 文件系统类型支持文件及文件夹安全设置，而 FAT32 文件系统类型不支持，NTFS 支持单文件最多为单个磁盘分区的容量大小 2TB，而 FAT32 单个最大文件容量不能超过 4GB。

Windows 操作系统从设计层面来讲，主要用来管理电脑硬件与软件资源的程序，大致包括 5 个方面的管理功能：进程与处理机管理、作业管理、存储管理、设备管理、文件管理。Windows 操作系统从个人使用角度来讲，主要用于个人电脑办公、软件安装、上网冲浪、游戏、数据分析、数据存储等。

2.3 硬盘分区简介

学习 Windows、Linux 操作系统，需要了解硬盘设备，硬盘是电脑主要的存储媒介之一，硬盘要能够安装系统或者存放数据，必须进行分区和格式化，Windows 系统常见分区有 3 种：主磁盘分区、扩展磁盘分区、逻辑磁盘分区。

一块硬盘设备，主分区至少有 1 个，最多 4 个，扩展分区可以为 0，最多 1 个，且主分区和扩展分区总数不能超过 4 个，逻辑分区可以有若干个。在 Windows 下激活的主分区是硬盘的启动分区，它是独立的，也是硬盘的第一个分区，通常就是我们所说的 C 盘系统分区。

扩展分区是不能直接用的，它是以逻辑分区的方式来使用的，扩展分区可分成若干逻辑分区。它们的关系是包含关系，所有的逻辑分区都是扩展分区的一部分。

在 Windows 系统安装时，硬盘驱动器是通过磁盘 0、磁盘 1 来显示，其中磁盘 0 表示第一块硬盘，磁盘 1 表示第二块硬盘，然后在第一块硬盘磁盘 0 上进行分区，最多不能超过 4 个主分区，分区为 C、D、E、F。

硬盘接口是硬盘与主机系统间的连接部件，作用是在硬盘缓存和主机内存之间传输数据。不同的硬盘接口决定着硬盘与计算机之间的连接速度，在整个系统中，硬盘接口的优劣直接影响着程序运行快慢和系统性能好坏，常见的硬盘接口类型为 IDE(integrated drive electronics)、SATA(serial advanced technology attachment)、SCSI(small computer system interface)、SAS(serial attached SCSI)和光纤通道等。

IDE 接口硬盘多用于家用，部分也应用于传统服务器，SCSI、SAS 接口的硬盘则主要应用于服务器市场，而光纤通道主要用于高端服务器上，SATA 主要用于个人家庭办公电脑及低端服务器。

在 Linux 操作系统中，读者可以看到硬盘驱动器的第一块 IDE 硬盘接口的硬盘设备为 hda，SATA 硬盘接口的硬盘设备为 sda，主分区编号为 hda1-4 或者 sda1-4，逻辑分区从 5 开始。如果有第二块硬盘，主分区编号为 hdb1-4 或者 sdb1-4。

不管是 Windows 还是 Linux 操作系统，硬盘的总容量＝主分区的容量＋扩展分区的容量，而扩展分区的容量＝各个逻辑分区的容量之和。主分区也可成为引导分区，会被操作系统和主板认定为这个硬盘的第一个分区，所以 C 盘永远都是排在所有磁盘分区的第一的位置上。

MBR(master boot record)和 GPT(GUID partition table)是在磁盘上存储分区信息的两种不同方式。这些分区信息包含了分区从哪里开始的信息，这样操作系统才知道哪个扇区是属于哪个分区，以及哪个分区是可以用来启动操作系统的。

在磁盘上创建分区时，必须选择 MBR 或者 GPT，默认是 MBR，也可以通过其他方式修改为 GPT 方式。MBR 分区的硬盘最多支持 4 个主分区，如果想支持更多主分区，可以考虑使用 GPT 格式分区。

2.4　Linux 安装环境准备

要学好 Linux 这门技术，首先需安装 Linux 操作系统，Linux 操作系统安装是每个初学者的门槛。而安装 Linux 操作系统，最大的困惑莫过于给操作系统进行磁盘分区。

虽然目前各种发行版本的 Linux 已经提供了友好的图形交互界面，但很多初学者还是感觉无从下手，其原因主要是不清楚 Linux 的分区规定。

Linux 系统安装中规定，同样每块硬盘设备最多只能分 4 个主分区(其中包含扩展分区)构成，任何一个扩展分区都要占用一个主分区号码，也就是在一个硬盘中，主分区和扩展分区一共最多是 4 个。

为了让读者能将本书所有 Linux 技术应用于企业，本书案例以企业里主流 Linux 操作

系统 CentOS 为蓝本，目前主流 CentOS 发行版本为 CentOS 7.2。

读者在安装 CentOS 操作系统时，如果没有多余的计算机裸机设备，可以在 Windows 主机上安装 VMware Workstation 工具，该工具的用途可以在 Windows 主机上创建多个计算机裸机设备资源，包括 CPU、内存、硬盘、网卡、DVD 光驱、USB 接口、声卡，创建的多个计算机裸机设备共享 Windows 主机的所有资源。

读者在安装 CentOS 操作系统时，如果有多余的计算机裸机设备或者企业服务器，可以将 CentOS 系统直接安装在服务器设备上，安装系统之前需要下载 CentOS 7.2 操作系统镜像文件(international organization for standardization，ISO 9660 标准)，通过刻录工具，将 ISO 镜像文件刻录至 DVD 光盘或者 U 盘里，通过 DVD 或者 U 盘启动，然后安装系统。

以下为在 Windwos 主机上安装 VMware Workstation 虚拟机软件，虚拟机软件的用途可以在真实机上模拟一台新的计算机资源，进而可以在新的计算机资源设备上安装 CentOS 7.2 操作系统，具体步骤如下。

(1) 安装环境准备。

- VMware Workstation 10.0；
- CentOS 7.2 x86_64。

(2) VMware Workstation 10.0 下载。

```
http://download3.vmware.com/software/wkst/file/VMware-workstation-full-10.0.1-1379776.exe
```

(3) CentOS 7.2 操作系统 ISO 镜像下载。

```
http://124.205.69.165/files/706900000291EB25/mirror.math.princeton.edu/pub/CentOS/7/isos/x86_
64/CentOS-7-x86_64-DVD-1511.iso
```

(4) 将 VMware Workstation 10.0 和 CentOS 7.2 ISO 镜像文件下载至 Windows 系统，双击 VMware-workstation-full-10.0.1-1379776.exe，根据提示完成安装，会在 Windows 桌面显示 VMware Workstation 图标，如图 2-2 所示。



图 2-2 VMware Workstation 图标

(5) 双击桌面 VMware Workstation 图标打开虚拟机软件,选择"创建新的虚拟机",如图 2-3 所示。



图 2-3 VMware Workstation 创建虚拟机

(6) 新建虚拟机向导,选择"自定义(高级)(C)"选项,如图 2-4 所示。

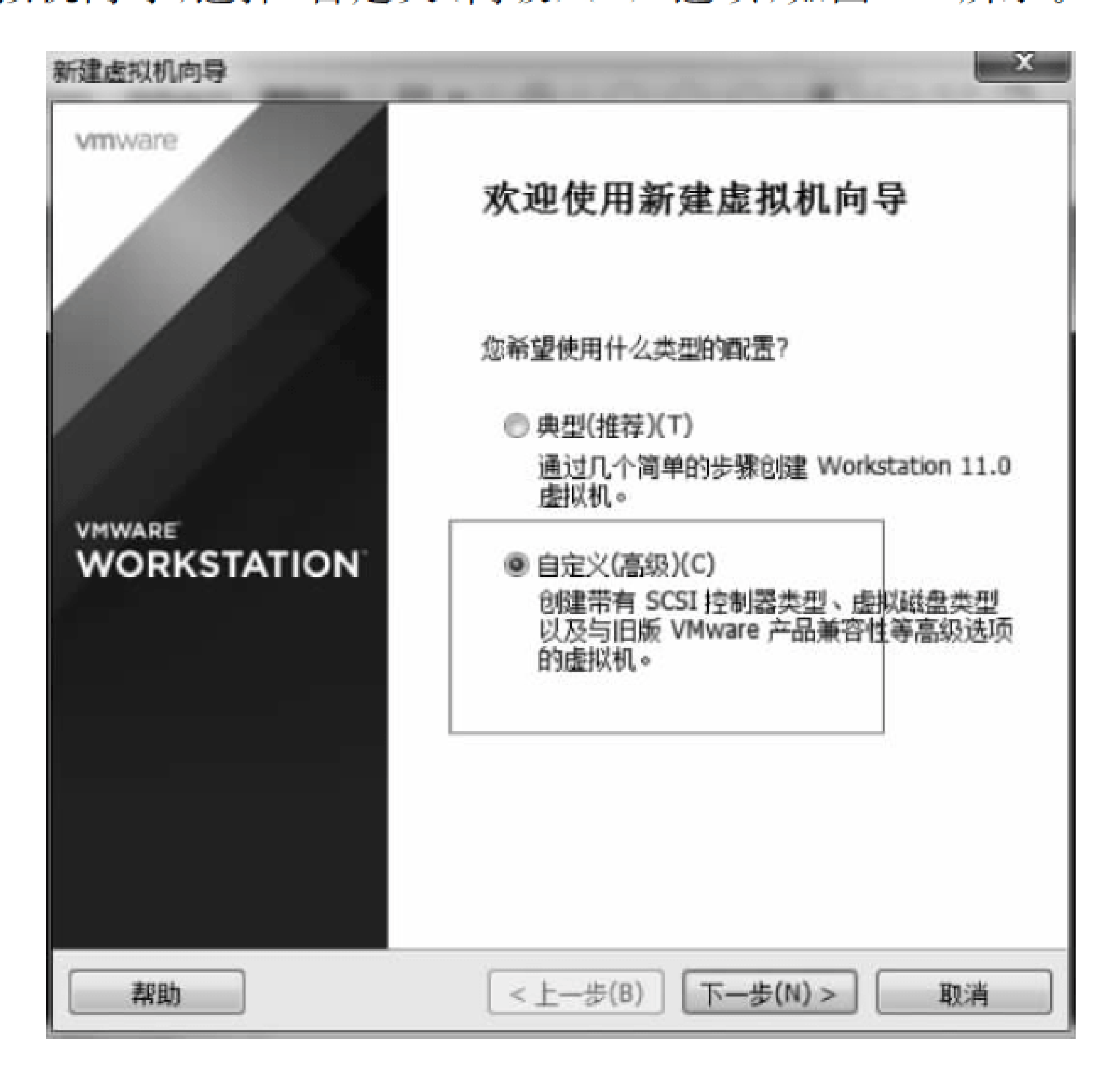


图 2-4 创建虚拟机向导

(7) 安装客户机操作系统,选择"稍后安装操作系统(S)",如图 2-5 所示。

(8) 选择客户机操作系统,由于即将安装 CentOS 7.2 操作系统,所以需要选择 Linux(L),同时版本(V)选择"CentOS 64 位",如图 2-6 所示。



图 2-5　安装客户机操作系统



图 2-6　操作系统版本

（9）虚拟机内存设置，默认为 1024MB，如图 2-7 所示。

（10）选择虚拟机网络类型，此处选择“使用桥接网络(R)”模式，如图 2-8 所示。



图 2-7 虚拟机内存分配

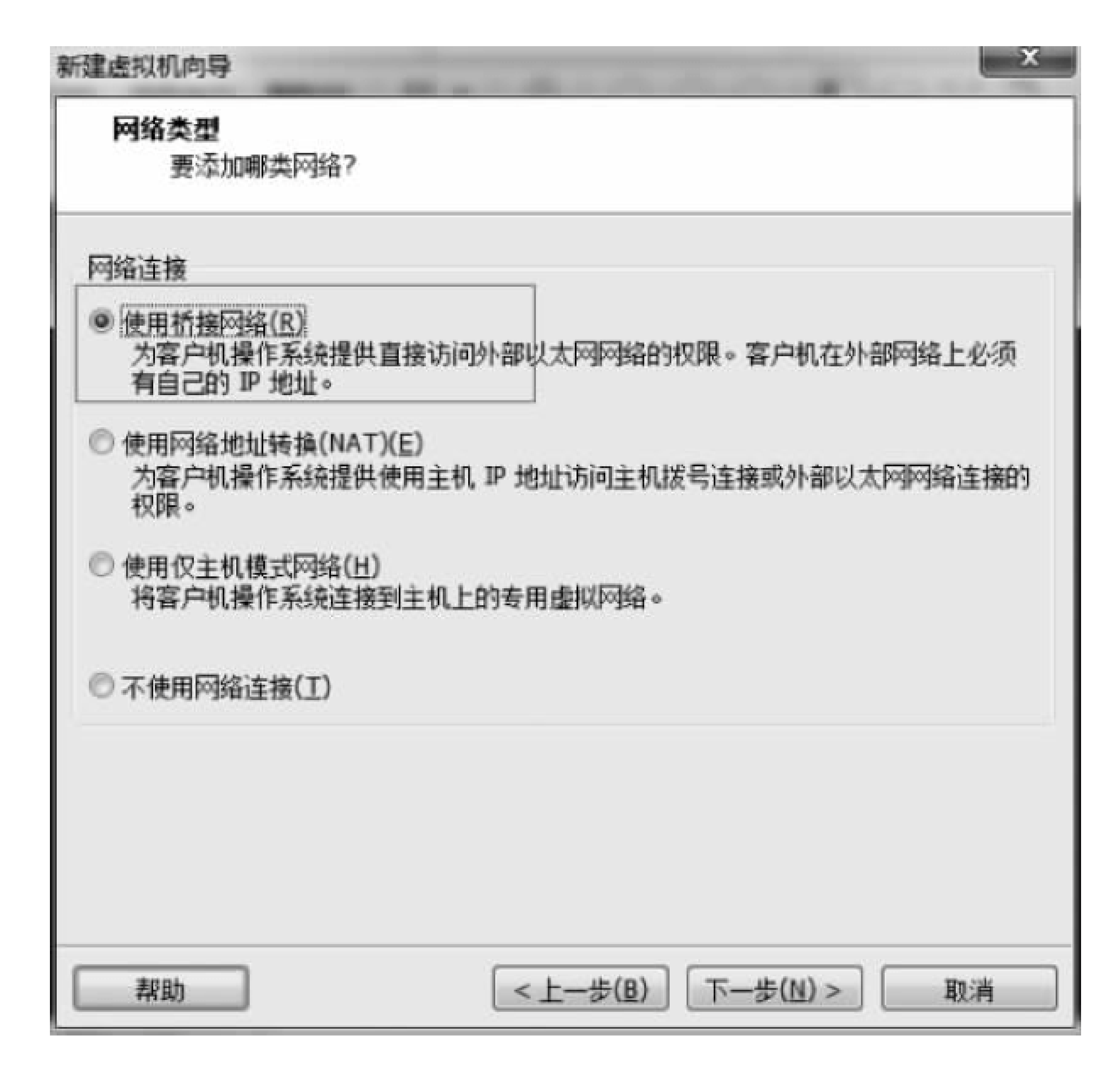


图 2-8 虚拟机网络类型

(11) 指定磁盘容量,设置虚拟机硬盘大小为 40GB,将虚拟磁盘拆分成多个文件,如图 2-9 所示。

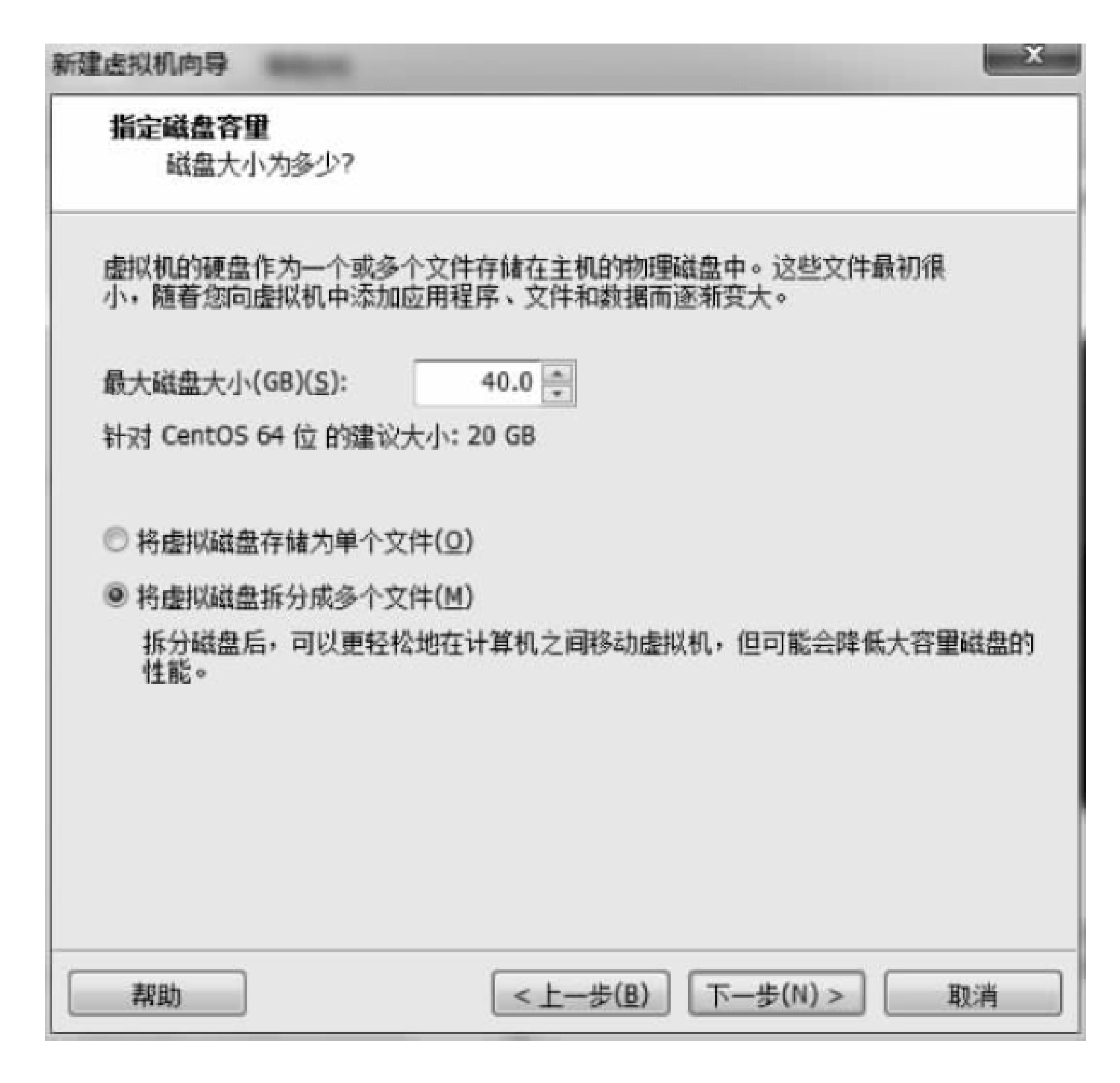


图 2-9　设置虚拟机磁盘容量

（12）虚拟机硬件资源创建完成，设备详情里面包括计算机常用设备，如内存、处理器、硬盘、CD/DVD、网络适配器等，如图 2-10 所示。



图 2-10　虚拟机裸机设备

（13）将 CentOS 7.2 ISO 系统镜像文件添加至虚拟机 CD/DVD 中，双击虚拟机“CD/DVD(IDE) 自动检测”选项，选择 CentOS-7-x86_64-DVD-1511.iso 镜像文件，如图 2-11 所示。

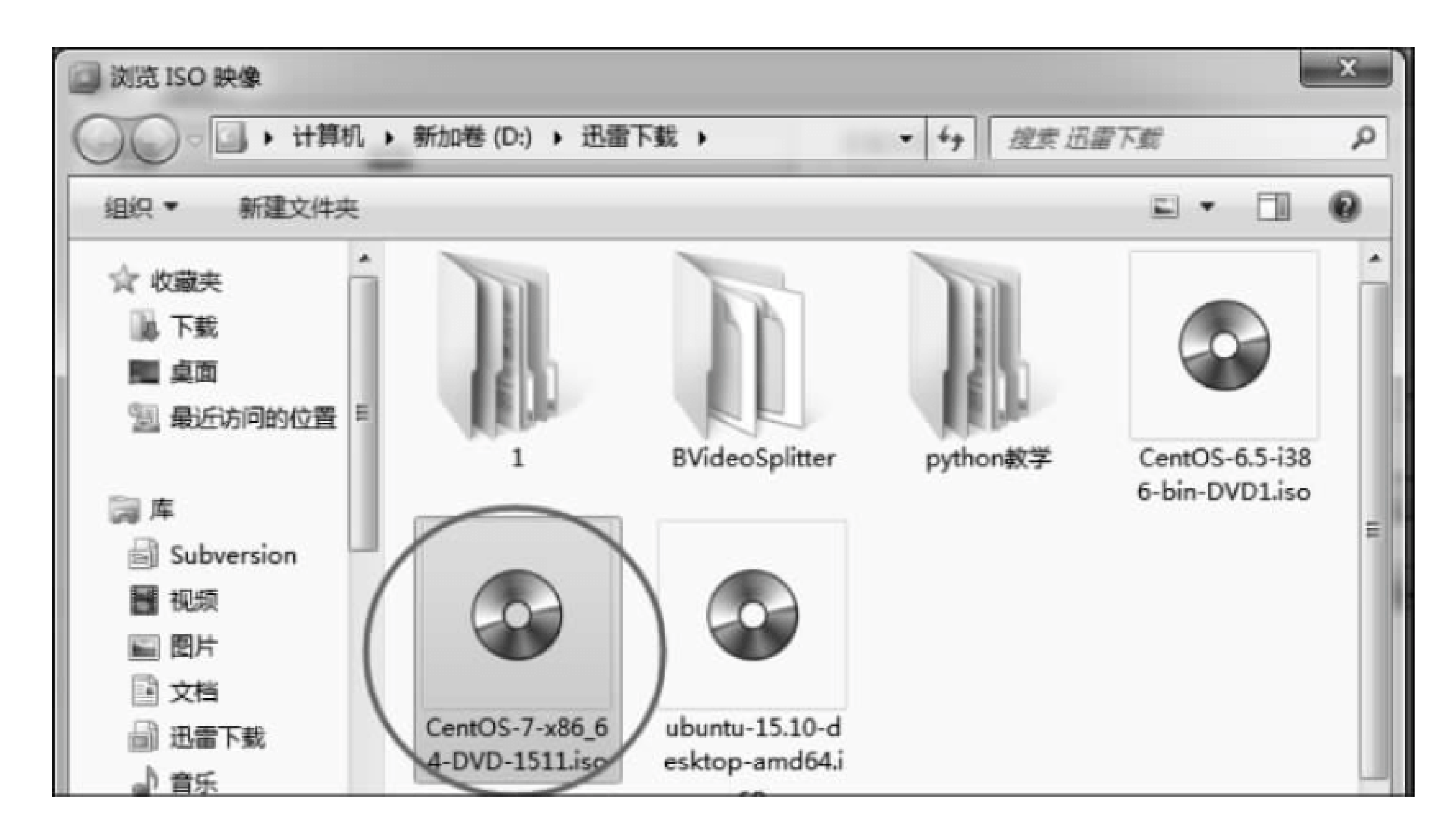


图 2-11 选择系统安装镜像

2.5 Linux 系统安装图解

如果直接在硬件主机设备上安装 CentOS 系统,则不需要安装虚拟机软件,直接插入 ISO 镜像 U 盘或者将镜像光盘插入 DVD 光驱即可,打开电源设备。具体安装步骤如下。

(1) 如图 2-12 所示,光标选择 Install CentOS 7,直接按 Enter 键进行系统安装。



图 2-12 选择安装菜单

(2) 继续按 Enter 键启动安装进程,进入光盘检测,按 Esc 键跳过检测,如图 2-13 所示。

(3) CentOS 7 欢迎界面,选择安装过程中界面显示的语言,初学者可以选择"简体中文"或者默认 English,如图 2-14 所示。

(4) CentOS 7 INSTALLATION SUMMARY 安装总览界面,如图 2-15 所示。

```
 -  Press the <ENTER> key to begin the installation process.

[  OK  ] Started Show Plymouth Boot Screen.
[  OK  ] Reached target Paths.
[  OK  ] Reached target Basic System.
[    7.169044] sd 2:0:0:0: [sda] Assuming drive cache: write through
[    7.259535] dracut-initqueue[671]: mount: /dev/sr0 is write-protected, mounting read-only
[  OK  ] Started Show Plymouth Boot Screen.
[  OK  ] Reached target Paths.
[  OK  ] Reached target Basic System.
[    7.259535] dracut-initqueue[671]: mount: /dev/sr0 is write-protected, mounting read-only
[  OK  ] Created slice system-checkisomd5.slice.
         Starting Media check on /dev/sr0...
/dev/sr0:   47d6f1bdfe9ab61a3b7a9a7227639841
Fragment sums: cc495d9e136c81ead92f382ef2f9d59118269d277a5cd55b91f6368991c1
Fragment count: 20
Press [Esc] to abort check.
Checking: 001.0%
```

图 2-13　跳过 ISO 镜像检测

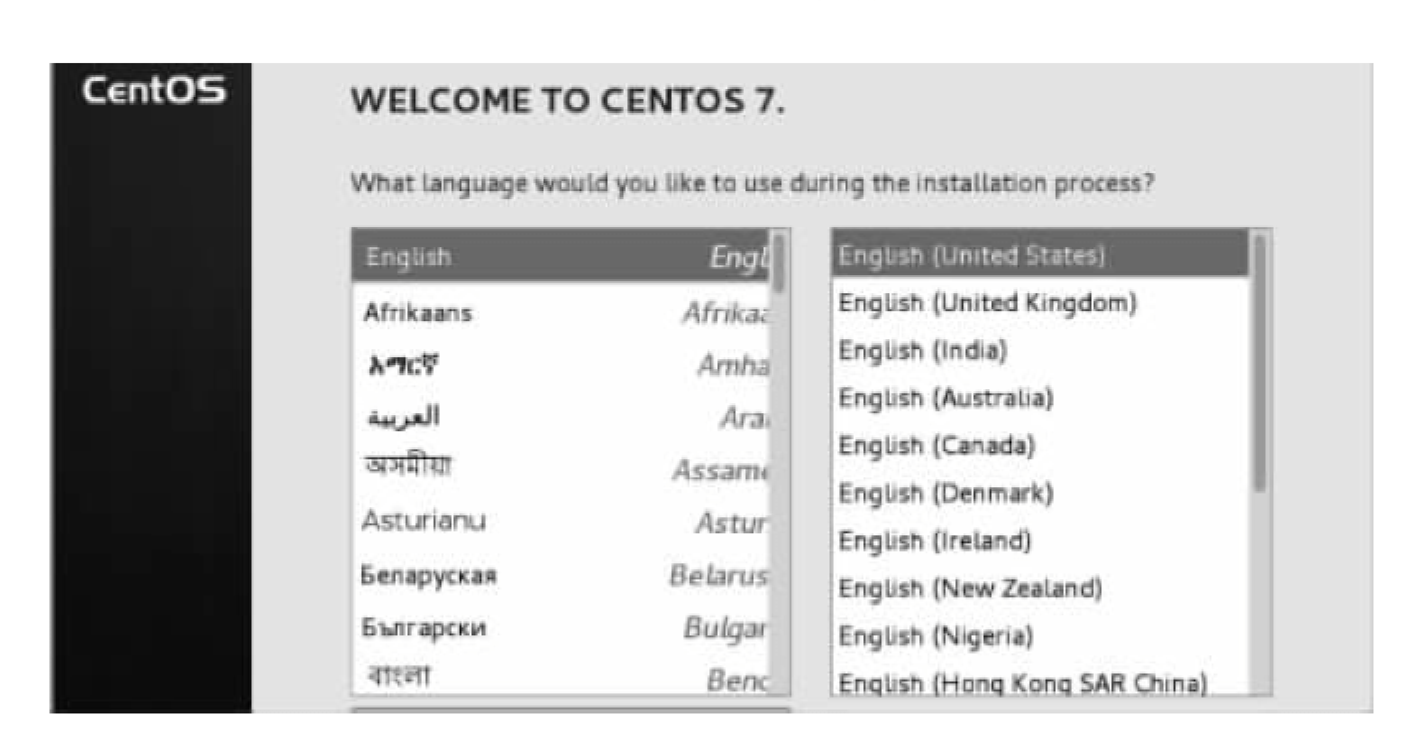
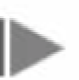


图 2-14　选择安装过程语言



图 2-15　INSTALLATION SUMMARY 界面



（5）选择 I will configure partitioning，并单击Done 按钮，如图 2-16 所示。

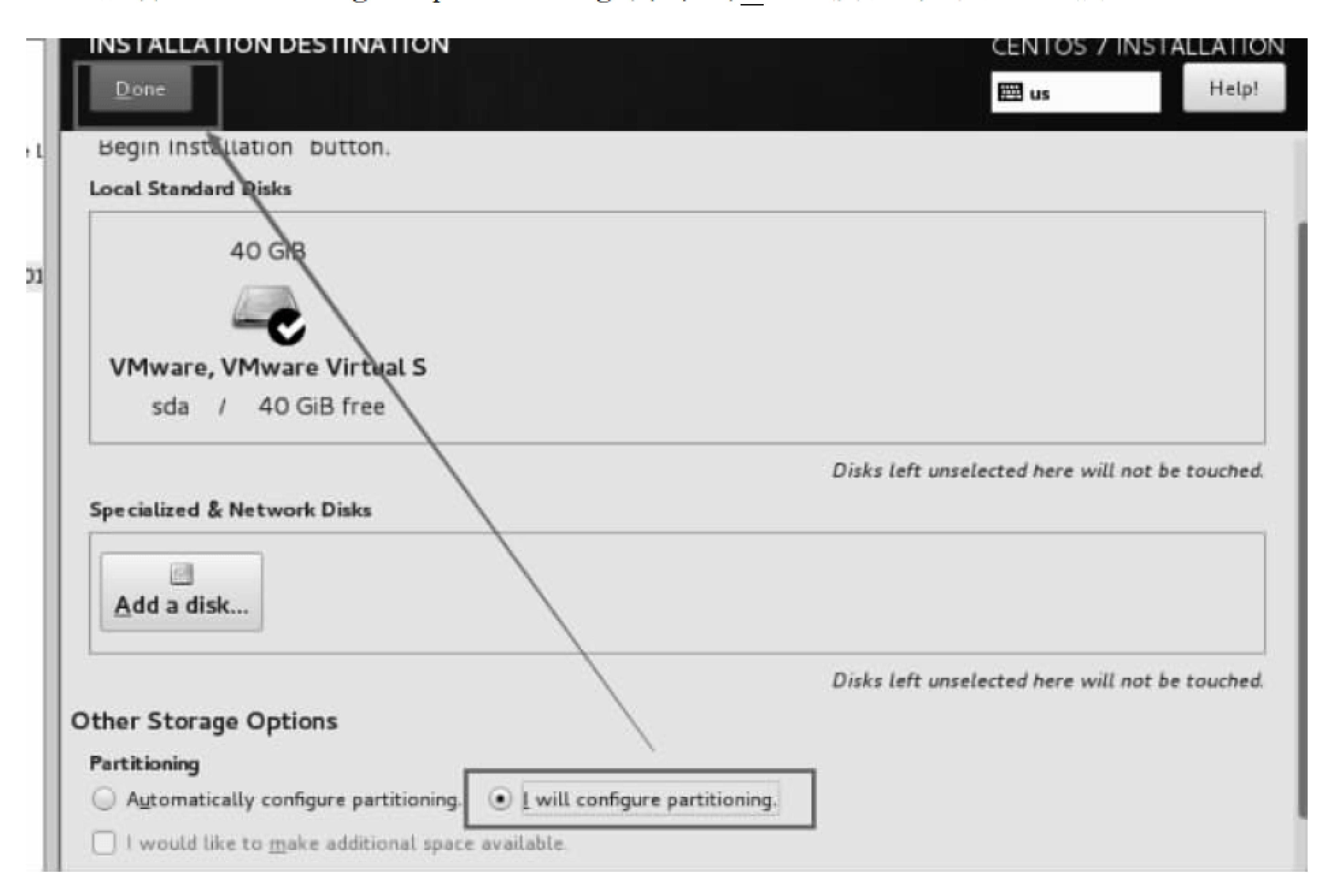


图 2-16　磁盘分区方式选择

（6）下拉框选择 Standard Partition，选择“＋”号，创建分区，如图 2-17 所示。

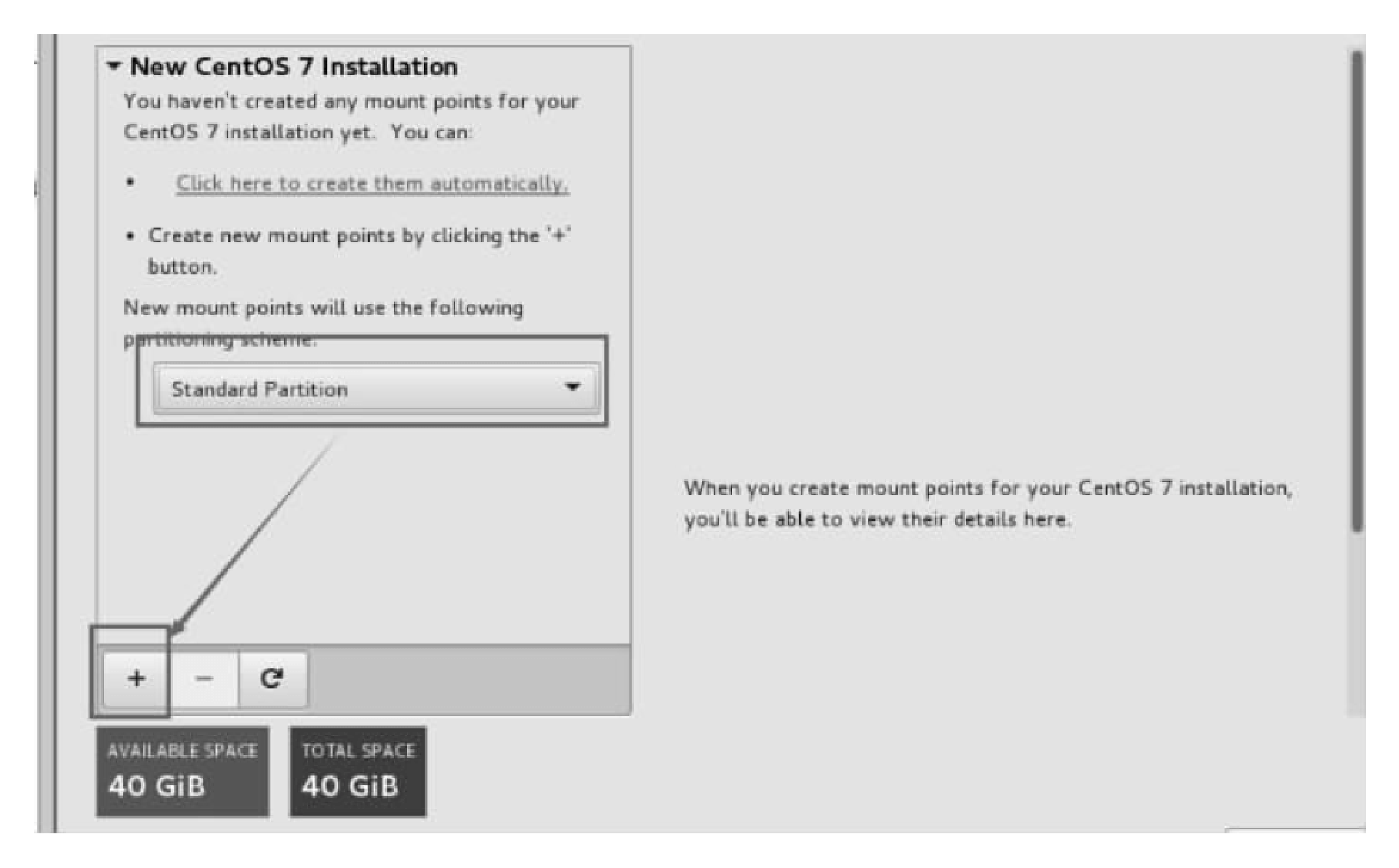


图 2-17　磁盘分区类型选择

（7）Linux 操作系统分区与 Windows 操作系统分区 C 盘、D 盘有很大区别，Linux 操作系统是采用树形的文件系统管理方式，所有的文件存储都以“/（根）”开始，如图 2-18 所示。

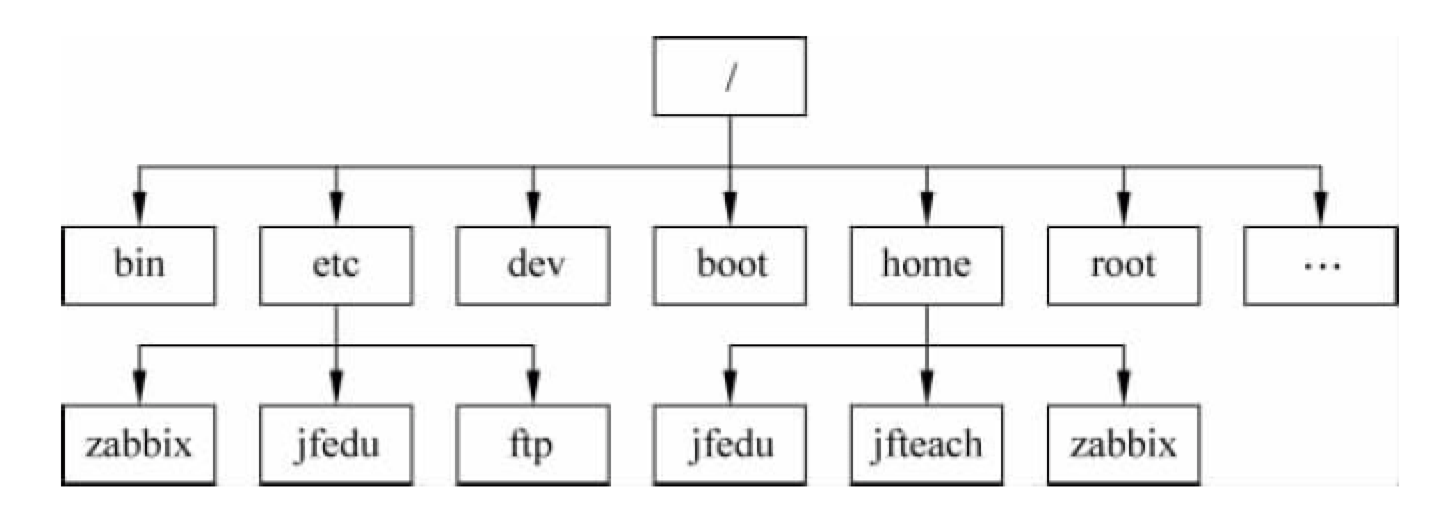


图 2-18　Linux 文件系统目录结构

Linux 是以文件的方式存储，例如/dev/sda 代表整块硬盘，/dev/sda1 表示硬盘第一分区，/dev/sda2 表示硬盘第二分区，为了能将目录和硬盘分区关联，所以 Linux 采用挂载点的方式来关联磁盘分区，/boot 目录、/根目录、/data 目录跟磁盘管理后，称之为分区，每个分区功能如下：

- /boot 分区用于存放 Linux 内核及系统启动过程所需文件；
- swap 分区又称为交换分区，类似 Windows 系统的虚拟内存，物理内存不够用时，以供程序使用 swap；
- /分区用于系统安装核心分区及所有文件存放的根系统；
- /data 分区为自定义分区，企业服务器中用于存放应用数据。

如图 2-19 所示，创建/boot 分区并挂载，分区大小为 200MB。

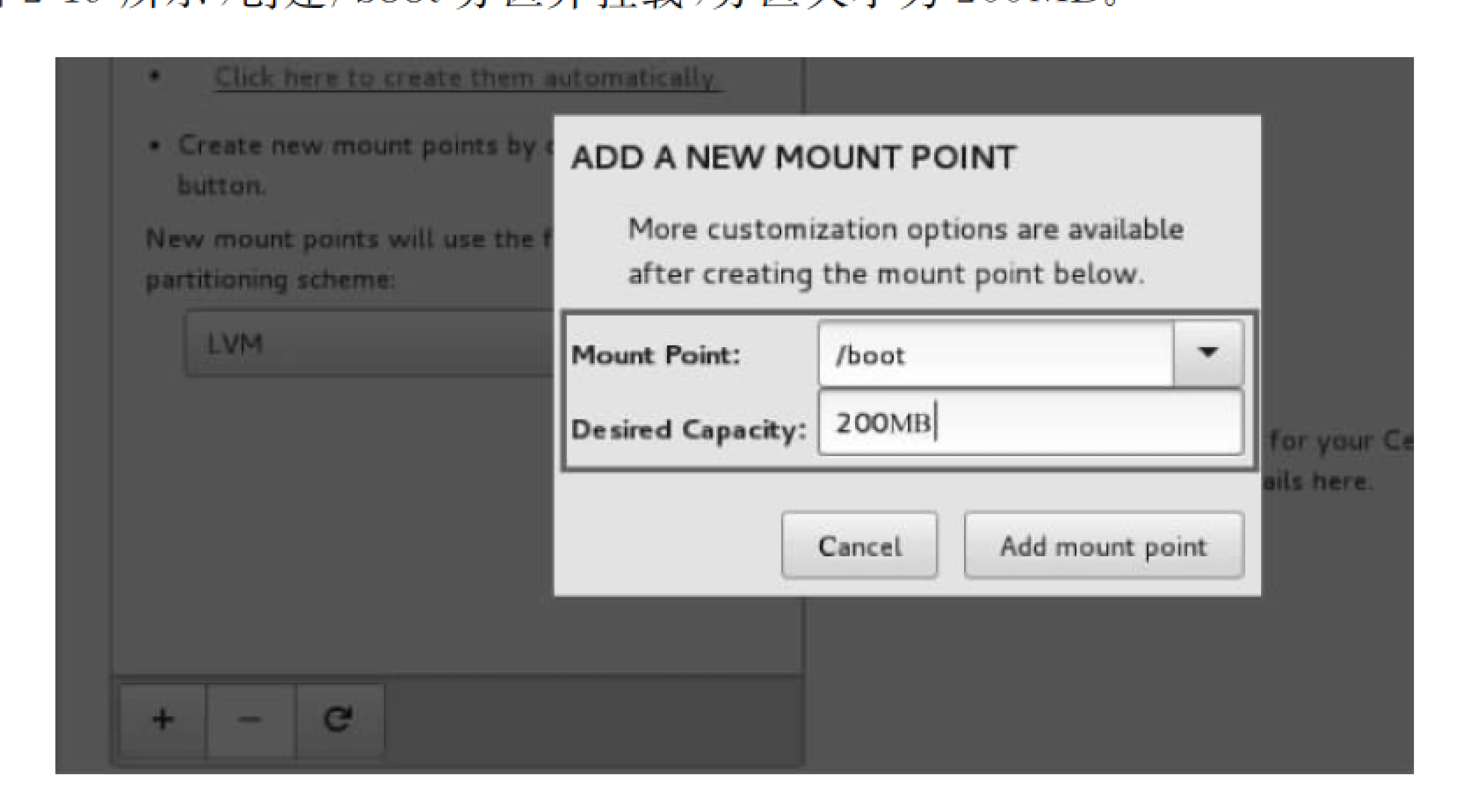


图 2-19　创建/boot 分区

单击 Add mount point 即可，磁盘分区默认文件系统类型为 XFS，根据上述方法，依次创建 swap 分区，大小为 2048MB，创建/分区，大小为剩余所有空间，最终如图 2-20 所示。

（8）选择 SOFTWARE SELECTION，设置为 Minimal Install 最小化安装，如果后期需

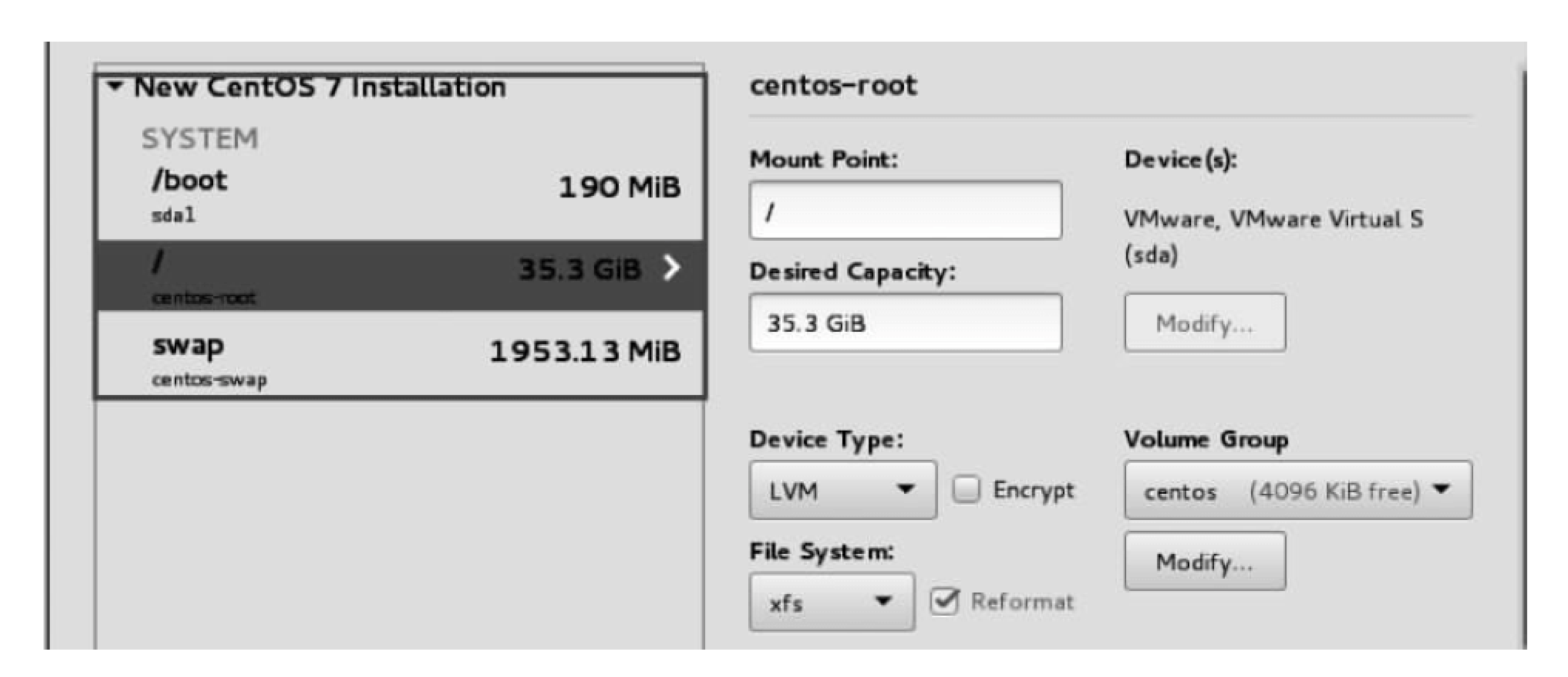


图 2-20 磁盘完整分区

要开发包、开发库等软件，可以在系统安装完后，根据需求安装即可，如图 2-21 所示。

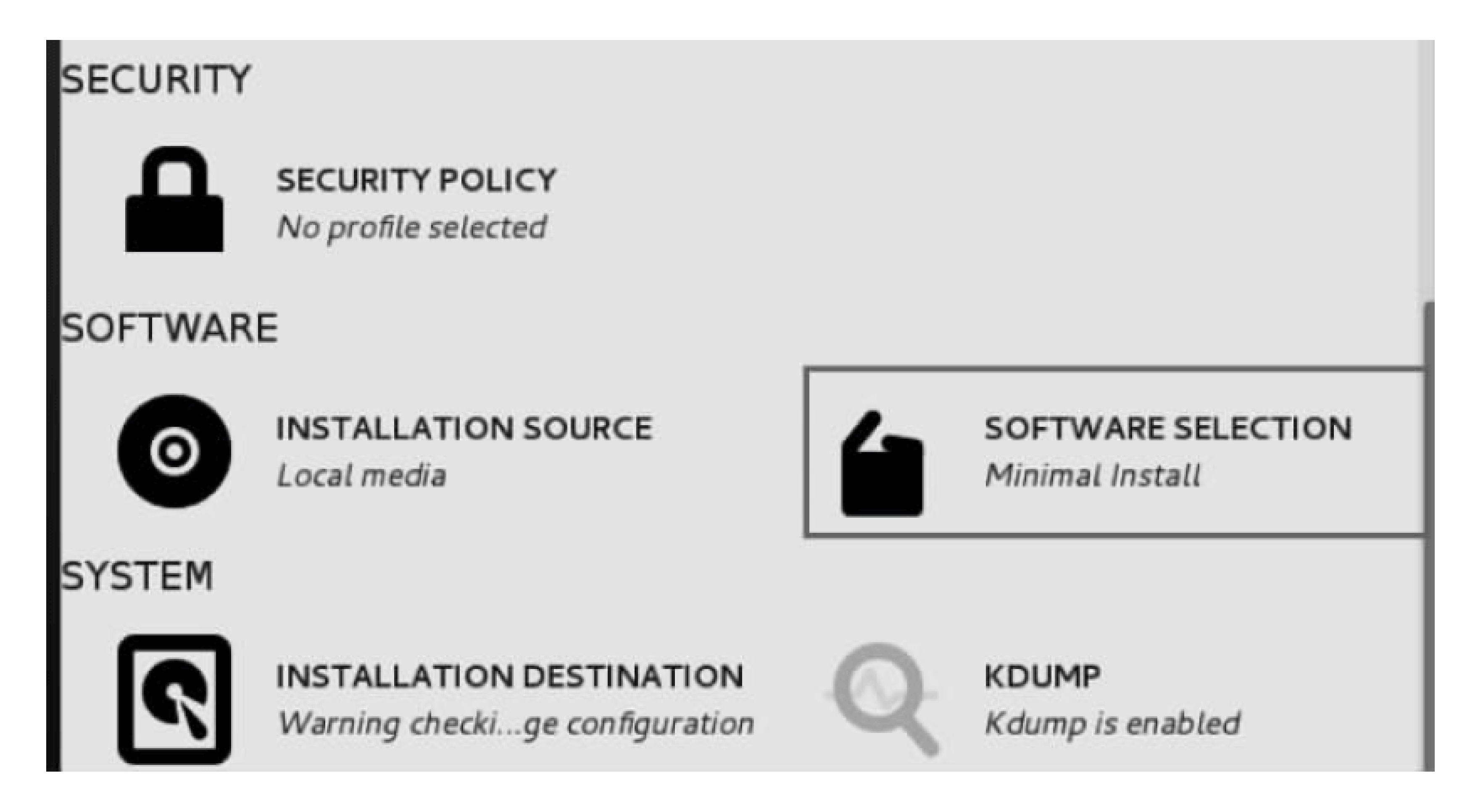


图 2-21 选择安装的软件

（9）操作系统时区选择，选择 Asia→Shanghai，关闭 Network Time，如图 2-22 所示。

（10）以上配置完毕后，单击 Begin Installation 后，单击 ROOT PASSWORD 设置 root 用户密码，如图 2-23 所示，如果需要新增普通用户，可以单击 USER CREATEION 进行创建即可。

（11）安装进程完毕，单击 Reboot 重启系统，如图 2-24 所示。

（12）重启 CentOS 7 Linux 操作系统，进入 login 登录界面，在"localhost login:"处输入 root，按 Enter 键，然后在"Password: "处输入系统安装时设定的密码，输入密码时不会提示，密码输入完按 Enter 键，即可登录 CentOS 7 Linux 操作系统，默认登录的终端称为 shell 终端，所有的后续操作指令均在 shell 终端上执行，默认显示字符提示[root@localhost ~]#，

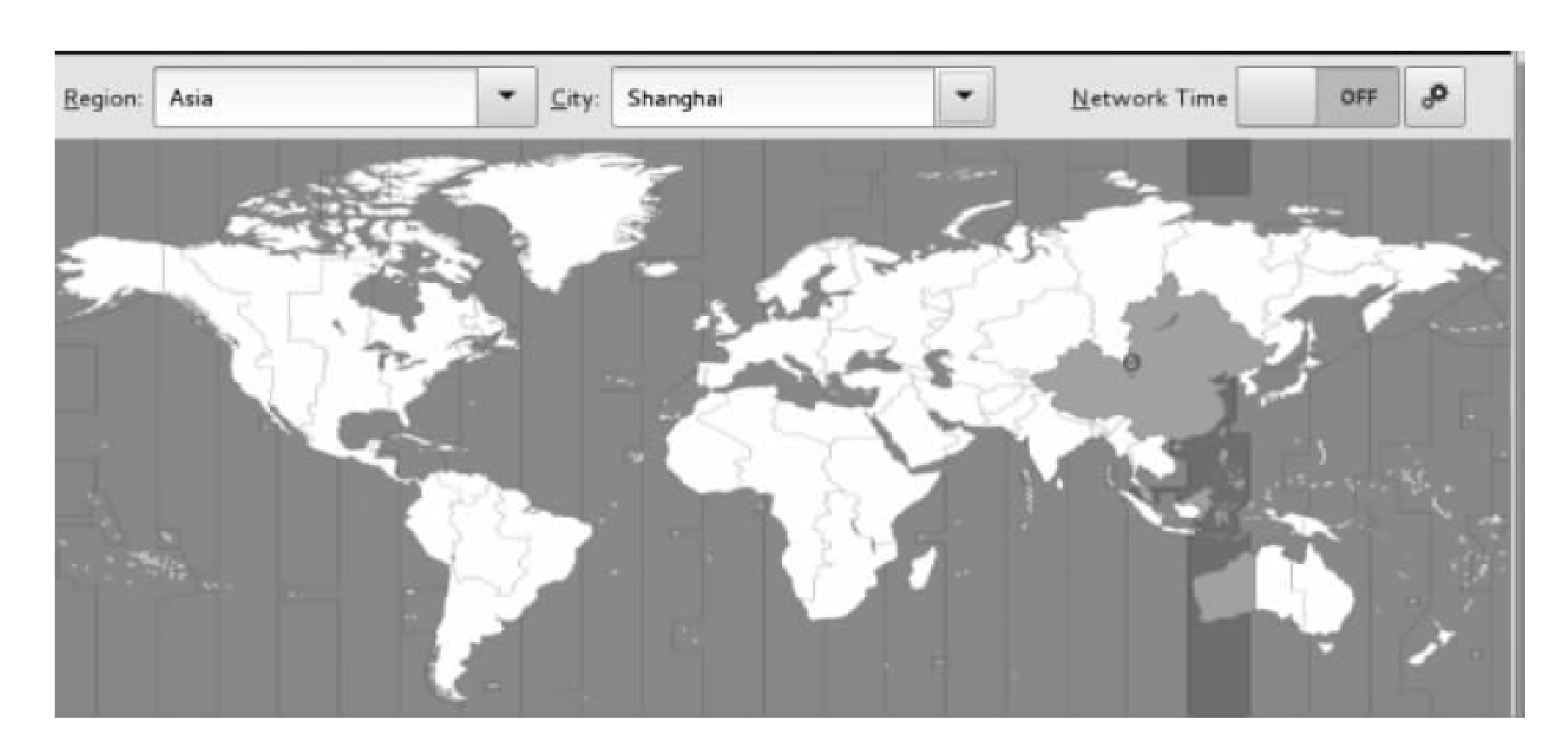


图 2-22 操作系统时区选择



图 2-23 设置 root 用户密码



图 2-24 系统安装完毕

其中“#”代表当前是 root 用户登录，如果是“$”表示当前为普通用户登录，如图 2-25 所示。

```
CentOS Linux 7 (Core)
Kernel 3.10.0-327.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Thu Aug 18 12:25:10 on tty1
[root@localhost ~]#
[root@localhost ~]# cat /etc/redhat-release
CentOS Linux release 7.2.1511 (Core)
[root@localhost ~]# uname -a
Linux localhost.localdomain 3.10.0-327.el7.x86_64 #1 SMP Thu Nov 19 22:10:57 UTC 2015 x86_64
[root@localhost ~]# pwd
/root
[root@localhost ~]# ls
anaconda-ks.cfg
```

图 2-25　login 登录系统界面

2.6　菜鸟学好 Linux 大绝招

Linux 系统安装是初学者的门槛，系统安装完毕后，很多初学者不知道该如何学习，不知道如何快速进阶，下面总结了菜鸟学好 Linux 技能的大绝招：

- 初学者完成 Linux 系统硬盘分区及安装之后，需熟练掌握 Linux 系统管理必备命令，命令包括：cd、ls、pwd、clear、chmod、chown、chattr、useradd、userdel、groupadd、vi、vim、cat、more、less、mv、cp、rm、rmdir、touch、ifconfig、ip addr、ping、route、echo、wc、expr、bc、ln、head、tail、who、hostname、top、df、du、netstat、ss、kill、alias、man、tar、zip、unzip、jar、fdisk、free、uptime、lsof、lsmod、lsattr、dd、date、crontab、ps、find、awk、sed、grep、sort、uniq 等，每个命令至少练习 30 遍，逐步掌握每个命令的用法及应用场景。
- 初学者进阶之路，需熟练构建 Linux 下常见服务：DHCP、SAMBA、DNS、Apache、MySQL、Nginx、Zabbix、Squid、Varnish、LVS、keepalived、ELK、MQ、Zookeeper、Docker、Openstack、Hbase、Mongodb、Redis 等，遇到问题先思考，没有头绪可以借助百度、Google 搜索引擎，问题解决后，将解决问题的步骤总结并形成文档。
- 理解操作系统的每个命令，每个服务的用途，为什么要配置这个服务，为什么需要调整该参数，只有带着目标去学习才能更快地成长，才能让你掌握更多新知识。
- 熟练搭建 Linux 系统上各种服务之后，需要理解每个服务的完整配置和优化，可以拓展思维。例如 LAMP 所有服务放在一台机器上，能否分开放在多台服务器以平衡压力呢，该如何去构建和部署呢？一台物理机构建 Docker 虚拟化，如果是 100 台、1000 台如何去实施呢，会遇到哪些问题呢？
- Shell 是 Linux 最经典的命令解释器，shell 脚本可以实现自动化运维，平时多练习 shell 脚本编程，每个 shell 脚本多练习几遍，从中吸取关键的参数、语法，不断地练习，不断地提高。
- 建立个人学习博客，把平时工作、学习中的知识都记录到博客，一方面可以供别人参

考，另一方面可以提高自己文档编写及总结的能力。

- 学习 Linux 技术是一个长期的过程，一定要坚持，遇到各种错误、问题可以借助百度、Google 搜索引擎，如果解决不了，可以请教同学、朋友及老师。
- 通过以上学习方法，不断进步，如果想达到高级、资深大牛级别，还需要进一步深入学习 Web 集群架构、网站负载均衡、网站架构优化、自动化运维、运维开发、虚拟化等知识。
- 多练习才是硬道理，实践出真知。

本章小结

通过对本章内容的学习，读者对 Linux 系统有了一个初步的理解，了解 Linux 行业的发展前景，学会了如何在企业中或者虚拟机中安装 Linux 系统。

对 32 位、64 位 CPU 处理器以及对 Linux 内核版本命名也有了进一步的认识，同时掌握了学习 Linux 的大绝招。

同步作业

1. 企业中服务器品牌 DELL R730，其硬盘总量为 300GB，现需安装 CentOS 7 Linux 操作系统，请问如何进行分区？
2. GNU 与 GPL 的区别是什么？
3. 企业一台 Linux 服务器，查看该 Linux 内核显示：3.10.0-327.36.3.el7.x86_64，请分别说出点号分割的每个数字及字母的含义？
4. CentOS Linux 至今发布了多少个系统版本？
5. 如果 Linux 系统采用光盘安装，如何将 ISO 镜像文件刻录成光盘，请写出具体实现流程。

第 3 章



CentOS 系统管理

Linux 系统安装完毕，需要对 Linux 系统进行管理和维护，让 Linux 服务器能真正应用于企业中。

本章向读者介绍 Linux 系统引导原理、启动流程、系统目录、权限、命令及 CentOS 7 和 CentOS 6 在系统管理、命令方面的区别等内容。

3.1 操作系统启动概念

不管是 Windows 还是 Linux 操作系统，底层设备一般均为物理硬件，操作系统启动之前会对硬件进行检测，然后硬盘引导启动操作系统，以下为与操作系统启动相关的几个概念。

3.1.1 BIOS

基本输入输出系统(basic input output system，BIOS)是一组固化到计算机主板上的只读内存镜像(read only memory image，ROM)芯片上的程序，它保存着计算机最重要的基本输入输出的程序、系统设置信息、开机后自检程序和系统自启动程序。主要功能是为计算机提供最底层的、最直接的硬件设置和控制。

3.1.2 MBR

全新硬盘在使用之前必须进行分区格式化，硬盘分区初始化的格式主要有两种，分别为 MBR 格式和 GPT 格式。

如果使用 MBR 格式，操作系统将创建主引导记录扇区(master boot record，MBR)，MBR 位于整块硬盘的 0 磁道 0 柱面 1 扇区，主要功能是操作系统对磁盘进行读写时，判断分区的合法性以及分区引导信息的定位。

主引导扇区总共为 512 字节，MBR 只占用了其中的 446 个字节，另外的 64 个字节为硬盘分区表(disk partition table，DPT)，最后两个字节“55，AA”是分区的结束标志。

在 MBR 硬盘中，硬盘分区信息直接存储于主引导记录（MBR）中，同时主引导记录还存储着系统的引导程序，如表 3-1 所示。

表 3-1　MBR 分区表

0000～0088	MBR 主引导程序	主引导程序
0089～01BD	出错信息数据区	数据区
01BE～01CD	分区项 1（16 字节）	分区表
01CE～01DD	分区项 2（16 字节）	
01DE～01ED	分区项 3（16 字节）	
01EE～01FD	分区项 4（16 字节）	
01FE	55	结束标志
01FF	AA	

MBR 是计算机启动最先执行的硬盘上的程序，只有 512 字节大小，所以不能载入操作系统的核心，只能先载入一个可以载入计算机核心的程序，称为引导程序。

因为 MBR 分区标准决定了 MBR 只支持在 2TB 以下的硬盘，对于后面的多余空间只能浪费。为了支持能使用大于 2TB 硬盘空间，微软和英特尔公司在可扩展固件接口（extensible firmware interface，EFI）方案中开发了全局唯一的标识符（globally unique identifier，GUID），进而全面支持大于 2TB 硬盘空间在企业中使用。

3.1.3　GPT

全局唯一的标识符（globally unique identifier，GUID），正逐渐取代 MBR 成为新标准。它和统一的可扩展固件接口（unified extensible firmware interface，UEFI）相辅相成。UEFI 用于取代老旧的 BIOS，而 GPT 则取代老旧的 MBR。之所以称为“GUID 分区表”，是因为驱动器上的每个分区都有一个全局唯一的标识符。

在 GPT 硬盘中，分区表的位置信息储存在 GPT 头中。出于兼容性考虑，第一个扇区同样有一个与 MBR 类似的标记，叫作受保护的主引导记录（protected main boot record，PMBR）。

PMBR 的作用是当使用不支持 GPT 的分区工具时，整个硬盘将显示为一个受保护的分区，以防止分区表及硬盘数据遭到破坏，而其中存储的内容和 MBR 一样，之后才是 GPT 头。

GPT 优点支持 2TB 以上磁盘，如果使用 Fdisk 分区，最大只能建立 2TB 大小的分区，创建大于 2TB 的分区，需使用 parted，同时必须使用 64 位操作系统，MAC、Linux 系统都能支持 GPT 分区格式，Windows 7/8 64 位、Windows Server 2008 64 位支持 GPT。GPT 硬盘分区表内容如图 3-1 所示。

3.1.4　GRUB

GNU 项目的多操作系统启动程序（GRand unified bootloader，GRUB），可以支持多操

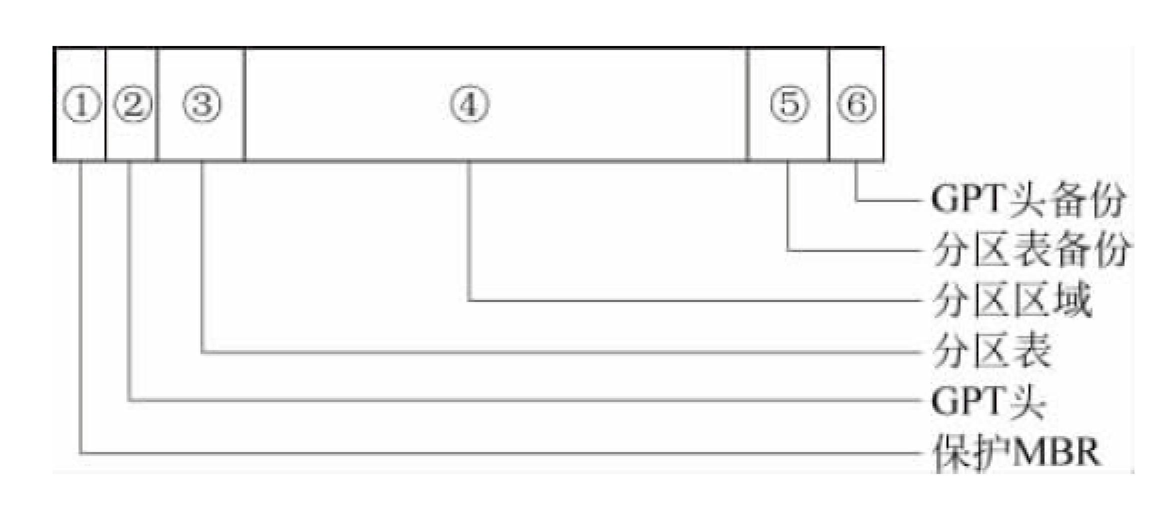


图 3-1　GPT 分区表内容

作系统的引导，它允许用户可以在计算机内同时拥有多个操作系统，并在计算机启动时选择希望运行的操作系统。

GRUB 可用于选择操作系统分区上的不同内核，也可用于向这些内核传递启动参数。它是一个多重操作系统启动管理器。用来引导不同系统，如 Windows、Linux。Linux 常见的引导程序包括 LILO、GRUB、GRUB2，CentOS 7 Linux 默认使用 GRUB2 引导程序，引导系统启动。如图 3-2 所示为 GRUB 加载引导流程。

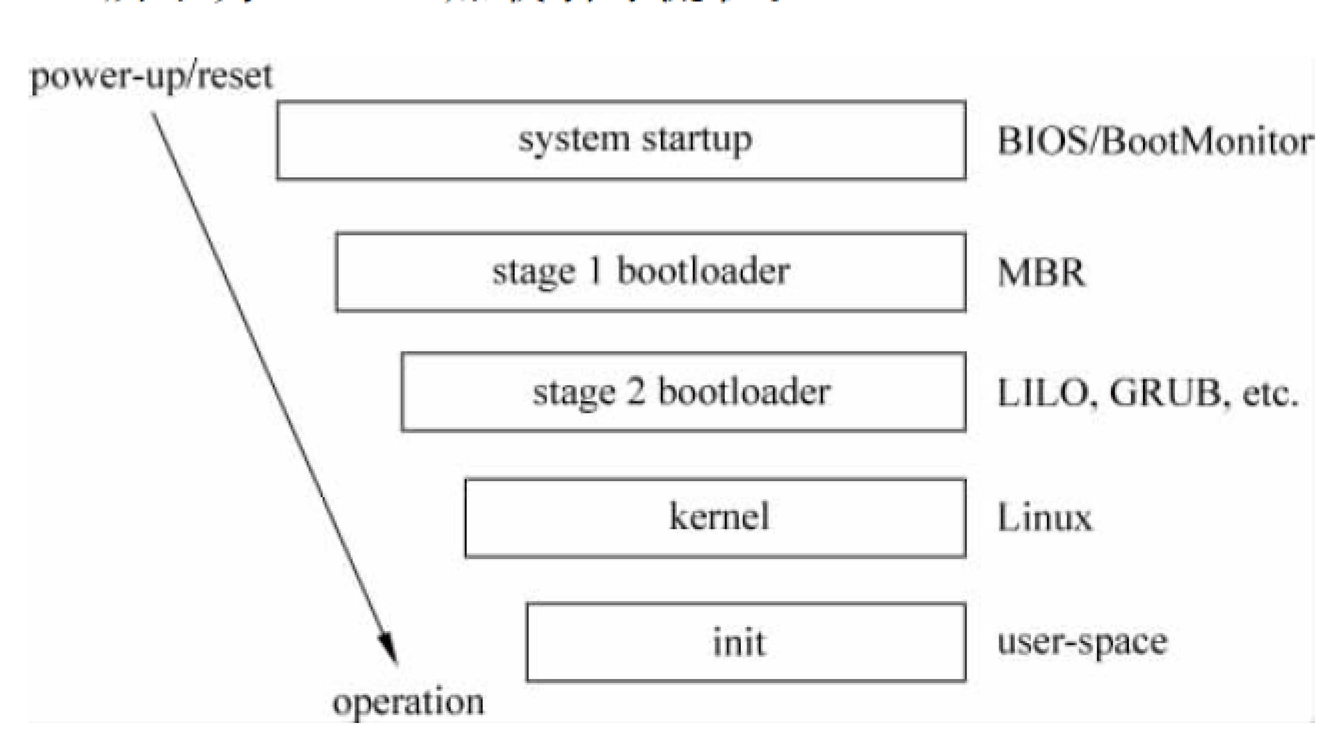


图 3-2　GRUB 引导流程

GRUB2 是基于 GRUB 开发成更加安全强大的多系统引导程序，最新 Linux 发行版都是使用 GRUB2 作为引导程序。同时 GRUB2 采用了模块化设计，使得 GRUB2 核心更加精炼，使用更加灵活，同时也就不需要像 GRUB 分为 stage 1、stage 1.5、stage 2 三个阶段。

3.2　Linux 操作系统启动流程

初学者对 Linux 操作系统启动流程深入理解，能有助于后期在企业中更好地维护 Linux 服务器，能够快速定位系统问题，进而解决问题。Linux 操作系统启动流程如图 3-3 所示。

1. 加载 BIOS

计算机电源加电质检，首先加载基本输入输出系统(basic input output system，BIOS)，

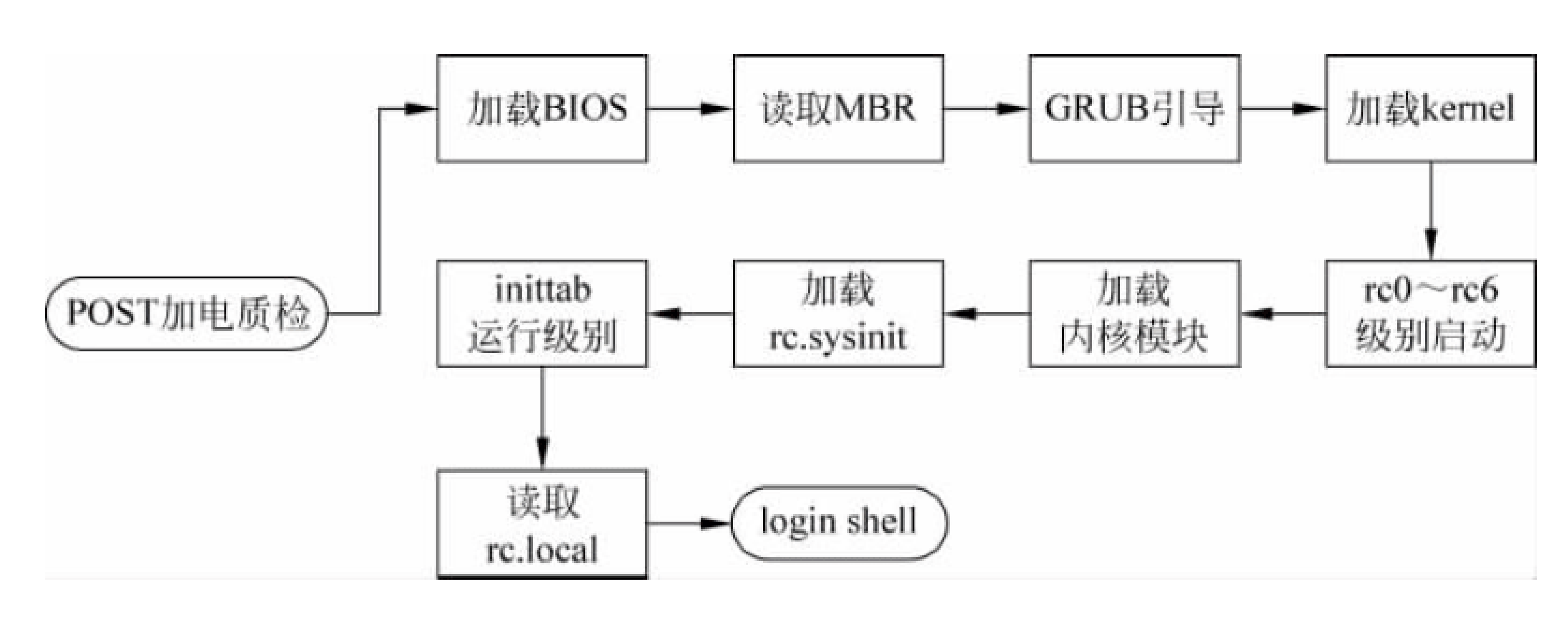


图 3-3 系统启动流程

BIOS 中包含硬件 CPU、内存、硬盘等相关信息，包含设备启动顺序信息、硬盘信息、内存信息、时钟信息、即插即用(plug-and-play，PNP)特性等。加载完 BIOS 信息，计算机将根据顺序进行启动。

2. 读取 MBR

读取完 BIOS 信息，计算机将会查找 BIOS 所指定的硬盘 MBR 引导扇区，将其内容复制到 0x7c00 地址所在的物理内存中。被复制到物理内存的内容是 bootloader，然后进行引导。

3. GRUB 引导

GRUB 启动引导器是计算机启动过程中运行的第一个软件程序，当计算机读取内存中的 GRUB 配置信息后，会根据其配置信息来启动硬盘中不同的操作系统。

4. 加载 kernel

计算机读取内存映像，并进行解压缩操作，屏幕一般会输出“Uncompressing Linux”的提示，当解压缩内核完成后，屏幕输出“OK，booting the kernel”。系统将解压后的内核放置在内存之中，并调用 start_kernel()函数来启动一系列的初始化函数并初始化各种设备，完成 Linux 核心环境的建立。

5. 设定 inittab 运行等级

内核加载完毕，会启动 Linux 操作系统第一个守护进程 init，然后通过该进程读取/etc/inittab 文件，/etc/inittab 文件的作用是设定 Linux 的运行等级，Linux 常见运行级别如下：

- 0：关机模式。
- 1：单用户模式。
- 2：无网络支持的多用户模式。
- 3：字符界面多用户模式。
- 4：保留，未使用模式。
- 5：图像界面多用户模式。
- 6：重新引导系统，重启模式。

6. 加载 rc. sysinit

读取完运行级别，Linux 系统执行的第一个用户层文件/etc/rc. d/rc. sysinit，该文件功

能包括设定 path 运行变量、设定网络配置、启动 swap 分区、设定/proc、系统函数、配置 SELinux 等。

7. 加载内核模块

读取/etc/modules. conf 文件及/etc/modules. d 目录下的文件来加载系统内核模块。该模块文件,可以后期添加或者修改及删除。

8. 启动运行级别程序

根据之前读取的运行级别,操作系统会运行 rc0. d 到 rc6. d 中的相应的脚本程序,来完成相应的初始化工作和启动相应的服务。其中以 S 开头表示系统即将启动的程序,如果以 K 开头,则代表停止该服务。S 和 K 后紧跟的数字为启动顺序编号,如图 3-4 所示。

```
[root@www-jfedu-net etc]# ls rc
rc          rc1.d/      rc3.d/      rc5.d/      rc.d/       rc.sysinit
rc0.d/      rc2.d/      rc4.d/      rc6.d/      rc.local
[root@www-jfedu-net etc]# ls rc5.d/
K01smartd              K16php-fpm             K75ntpdate             K99lvm2-mon
K05atd                 K35dovecot             K75quota_nld           K99rngd
K05trqauthd            K36mysqld              K80redis               S01sysstat
K10flow-capture        K50kdump               K85mdmonitor           S10network
K10psacct              K50netconsole          K85messagebus          S11portrese
K10saslauthd           K50snmpd               K87irqbalance          S12rsyslog
K10zabbix_server       K50snmptrapd           K87restorecond         S25cups
K15collectd            K60munge               K88auditd              S26udev-pos
K15htcacheclean        K61nfs-rdma            K89rdisc               S30nscd
K15httpd               K74acpid               K92ip6tables           S50aegis
K15svnserve            K74haldaemon           K92iptables            S50cloud-in
K16abrt-ccpp           K75blk-availability    K95rdma                S51cloud-in
K16abrtd               K75netfs               K99cpuspeed            S52cloud-in
[root@www-jfedu-net etc]# ls rc5.d/
```

图 3-4　运行级别服务

9. 读取 rc. local 文件

操作系统启动完相应服务之后,会读取执行/etc/rc. d/rc. local 文件,可以将需要开机启动的任务加入到该文件末尾,系统会逐行去执行并启动相应命令,如图 3-5 所示。

```
[root@www-jfedu-net ~]# cat /etc/rc.d/rc.local
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.
touch /var/lock/subsys/local
modprobe nf_conntrack
svnserve -d -r /data/svn/ --listen-port=8801
##############
/sbin/swapon /data/swapfile
/etc/init.d/httpd restart
/etc/init.d/mysqld restart
/usr/local/nginx/sbin/nginx
[root@www-jfedu-net ~]#
```

图 3-5　开机运行加载文件

10. 执行/bin/login 程序

执行/bin/login 程序,启动到系统登录界面,操作系统等待用户输入用户名和密码,即可登录到 shell 终端,如图 3-6 所示,输入用户名、密码即可登录 Linux 操作系统,至此 Linux 操作系统完整流程启动完毕。

```
CentOS Linux 7 (Core)
Kernel 3.10.0-327.el7.x86_64 on an x86_64

localhost login: root
Password:
```

图 3-6　系统登录界面

3.3　CentOS 6 与 CentOS 7 区别

CentOS 6 默认采用 sysvinit 风格，sysvinit 就是 system V 风格的 init 系统，sysvinit 用术语 runlevel 来定义“预订的运行模式”。sysvinit 检查“/etc/inittab”文件中是否含有“initdefault”项，该选项指定 init 的默认运行模式。sysvinit 使用脚本，文件命名规则和软链接来实现不同的 runlevel，串行启动各个进程及服务。

CentOS 7 默认采用 systemd 风格，systemd 是 Linux 系统中最新的初始化系统(init)，它主要的设计目标是克服 sysvinit 固有的缺点，提高系统的启动速度。

systemd 和 Ubuntu 的 upstart 是竞争对手，预计会取代 upstart。systemd 的目标是尽可能启动更少的进程，尽可能将更多进程并行启动。如表 3-2 所示为 CentOS 6 与 CentOS 7 操作系统的区别。

表 3-2　CentOS 6 与 CentOS 7 操作系统区别

A	B	C	D
编号	系统功能	CentOS 6	CentOS 7
1	init 系统	sysvinit	systemd
2	桌面系统	GNOME 2. X	GNOME 3. X/GNOME shell
3	文件系统	EXT4	XFS
4	内核版本	2. 6. X	3. 10. X
5	启动加载器	GRUB Legacy(＋efibootmgr)	GRUB2
6	防火墙	iptables	firewalld
7	数据库	MySQL	MariaDB
8	文件目录	/bin,/sbin,/lib,/lib64 在/根下	/bin,/sbin,/lib,/lib64 在/usr 下
9	主机名	/etc/sysconfig/network	/etc/hostname
10	时间同步	ntp,ntpq -p	chrony,chronyc sources

11	修改时间	# vi/etc/sysconfig/clock
		ZONE="Asia/Tokyo"
		UTC=false
		# ln -s
		/usr/share/zoneinfo/Asia/Tokyo
		/etc/localtime
			# timedatectl set-timezone Asia/Tokyo
			# timedatectl status

续表

A	B	C	D
编号	系统功能	CentOS 6	CentOS 7
12	区域及字符设置	/etc/sysconfig/i18n	/etc/locale. conf localectl set-locale LANG=zh_CN. utf8 localectl status
13	启动停止服务	# service service_name start # service service_name stop # service sshd restart/status/reload	# systemctl start service_name # systemctl stop service_name # systemctl restart/status/reload sshd
14	自动启动	chkconfig service_name on/off	# systemctl enable service_name # systemctl disable service_name
15	服务列表	chkconfig --list	# systemctl list-unit-files # systemctl --type service
16	Kill 服务	kill -9 < PID >	systemctl kill --signal=9 sshd
17	网络及端口信息	netstat	ss
18	IP 信息	ifconfig	ip address show
19	路由信息	route -n	ip route show
20	关闭停止系统	shutdown -h now	systemctl poweroff
21	单用户模式	init S	systemctl rescue
22	运行模式	vim/etc/inittab id:3:initdefault:	systemctl set-default graphical. target systemctl set-default multi-user. target

Linux 操作系统文件系统类型主要有 EXT3、EXT4、XFS 等，其中 CentOS 6 普遍采用 EXT3 和 EXT4 文件系统格式，而 CentOS 7 默认采用 XFS 格式。以下为 EXT3、EXT4、XFS 的区别：

- EXT4 是第四代扩展文件系统(fourth extended filesystem，EXT4)是 Linux 系统下的日志文件系统，是 EXT3 文件系统的后继版本；
- EXT3 类型文件系统支持最大 16TB 文件系统和最大 2TB 文件；
- EXT4 分别支持 1EB(1EB=1024PB，1PB=1024TB)的文件系统，以及 16TB 的单个文件；
- EXT3 只支持 32000 个子目录，而 EXT4 支持无限数量的子目录；
- EXT4 磁盘结构的 inode 个数支持 40 亿，而且 EXT4 的单个文件大小支持到 16TB；
- XFS 是一个 64 位文件系统，最大支持 8EB 减 1 字节的单个文件系统，实际部署时取决于宿主操作系统的最大块限制，常用于 64 位操作系统，发挥更好的性能；
- XFS 一种高性能的日志文件系统，最早于 1993 年，由 Silicon Graphics 为他们的 IRIX 操作系统而开发，是 IRIX 5. 3 版的默认文件系统；
- XFS 于 2000 年 5 月，Silicon Graphics 以 GPL 发布这套系统的源代码，之后被移植到 Linux 内核上，XFS 特别擅长处理大文件，同时提供平滑的数据传输。

3.4 TCP/IP 协议概述

要学好 Linux，对网络协议也要有充分的了解和掌握，例如传输控制协议/因特网互联协议(transmission control protocol/internet protocol，TCP/IP)，TCP/IP 名为网络通信协议，是 Internet 最基本的协议、Internet 国际互联网络的基础，由网络层的 IP 协议和传输层的 TCP 协议组成。

TCP/IP 定义了电子设备如何连入因特网，以及数据如何在它们之间传输的标准。协议采用了 4 层的层级结构，每一层都呼叫它的下一层所提供的协议来完成自己的需求。

TCP 负责发现传输的问题，一旦有问题就发出信号，要求重新传输，直到所有数据安全正确地传输到目的地，而 IP 是给因特网的每台联网设备规定一个地址。

基于 TCP/IP 的参考模型将协议分成 4 个层次，分别是网络接口层、网际互联层(IP 层)、传输层(TCP 层)和应用层。如图 3-7 所示为 TCP/IP 和 OSI 参考模型层次的对比。

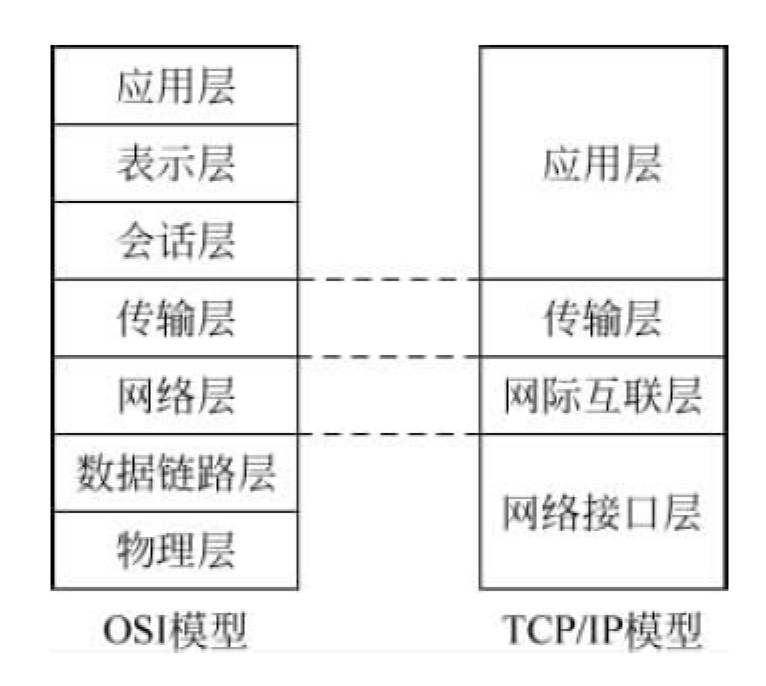


图 3-7 OSI 7 层模型与 TCP/IP 4 层对比

OSI 模型与 TCP/IP 模型协议功能实现对照表，如表 3-3 所示。

表 3-3 OSI 7 层模型与 TCP/IP 层次功能对比

OSI 7 层	每层功能	TCP/IP 协议簇
应用层	文件传输、电子邮件、文件服务、虚拟终端	TFTP、HTTP、SNMP、FTP、SMTP、DNS、Telnet
表示层	数据格式化、代码转换、数据加密	没有协议
会话层	解除或建立与别的接点的联系	没有协议
传输层	提供端对端的接口	TCP、UDP
网络层	为数据包选择路由	IP、ICMP、OSPF、BGP、IGMP、ARP、RARP
数据链路层	传输有地址的帧以及错误检测功能	SLIP、PPP、MTU
物理层	以二进制数据形式在物理媒体上传输数据	ISO 2110、IEEE 802、IEEE 802.2

3.5 IP 地址及网络常识

互联网协议地址(internet protocol address,IP),IP 地址是 IP 协议提供的一种统一的地址格式,它为互联网上的每一个网络和每一台主机分配一个逻辑地址,以此来屏蔽物理地址的差异。IP 地址给 Internet 上的每个通信设备分配一个编号,每台联网的 PC 上都需要有 IP 地址,这样才能正常通信。

IP 地址是一个 32 位的二进制数,通常被分割为 4 个 8 位二进制数(即 4 个字节)。IP 地址通常用点分十进制表示成“a. b. c. d”的形式,其中,a、b、c、d 都是 0～255 之间的十进制整数。

常见的 IP 地址分为 IPv4 与 IPv6 两大类。IP 地址编址方案将 IP 地址空间划分为 A、B、C、D、E 五类,其中 A、B、C 是基本类,D、E 类作为多播和保留使用。

IPv4 有 4 段数字,每一段最大不超过 255。由于互联网的蓬勃发展,IP 位址的需求量越来越大,使得 IP 位址的发放愈趋严格,各项资料显示,全球 IPv4 位址在 2011 年已经全部分发完毕。

地址空间的不足必将妨碍互联网的进一步发展。为了扩大地址空间,拟通过 IPv6 重新定义地址空间。IPv6 采用 128 位地址长度。在 IPv6 的设计过程中除了一劳永逸地解决了地址短缺问题以外,IPv6 的诞生可以给全球每一粒沙子配置一个 IP 地址,还考虑了在 IPv4 中解决不好的其他问题,如图 3-8 所示。

```
docker0   Link encap:Ethernet  HWaddr FA:C4:FD:2D:6A:39
          inet addr:10.0.42.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::f8c4:fdff:fe2d:6a39/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:465 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:19746 (19.2 KiB)
```

图 3-8 IPv4 与 IPv6 地址

3.5.1 IP 地址分类

IPv4 地址编址方案有 A、B、C、D、E 五类,其中 A、B、C 是基本类,D、E 类作为多播和保留使用,各分类详解如下。

1. A 类 IP 地址

一个 A 类 IP 地址是指,在 IP 地址的四段号码中,第一段号码为网络号码,剩下的三段号码为本地计算机的号码。如果用二进制表示 IP 地址的话,A 类 IP 地址就由 1 字节的网络地址和 3 字节主机地址组成,网络地址的最高位必须是 0。A 类 IP 地址中网络的标识长度为 8 位,主机标识的长度为 24 位,A 类网络地址数量较少,有 126 个网络,每个网络可以容纳主机数可达到 1600 万台。

A 类 IP 地址范围为 1. 0. 0. 0～127. 255. 255. 255 (二进制表示为 00000001 00000000

00000000 00000000～01111110 11111111 11111111 11111111)，最后一个为广播地址。A类 IP 地址的子网掩码为 255.0.0.0，每个网络支持的最大主机数为 $256^3-2=16777214$ 台。

2. B类 IP 地址

一个 B 类 IP 地址是指在 IP 地址的四段号码中，前两段号码为网络号码。如果用二进制表示 IP 地址的话，B 类 IP 地址就由 2 字节的网络地址和 2 字节主机地址组成，网络地址的最高位必须是 10。

B 类 IP 地址中网络的标识长度为 16 位，主机标识的长度为 16 位，B 类网络地址适用于中等规模的网络，有 16384 个网络，每个网络所能容纳的计算机数为 6 万多台。

B 类 IP 地址范围为 128.0.0.0～191.255.255.255(二进制表示为 10000000 00000000 00000000 00000000～10111111 11111111 11111111 11111111)，最后一个是广播地址。B 类 IP 地址的子网掩码为 255.255.0.0，每个网络支持的最大主机数为 $256^2-2=65534$ 台。

3. C类 IP 地址

一个 C 类 IP 地址是指在 IP 地址的四段号码中，前三段号码为网络号码，剩下的一段号码为本地计算机的号码。如果用二进制表示 IP 地址的话，C 类 IP 地址就由 3 字节的网络地址和 1 字节主机地址组成，网络地址的最高位必须是 110。C 类 IP 地址中网络的标识长度为 24 位，主机标识的长度为 8 位，C 类网络地址数量较多，有 209 万余个网络。适用于小规模的局域网络，每个网络最多只能包含 254 台计算机。

C 类 IP 地址范围为 192.0.0.0～223.255.255.255(二进制表示为 11000000 00000000 00000000 00000000～11011111 11111111 11111111 11111111)。C 类 IP 地址的子网掩码为 255.255.255.0，每个网络支持的最大主机数为 $256-2=254$ 台。

4. D类 IP 地址

D 类 IP 地址又称为多播地址(multicast address)，即组播地址。在以太网中，多播地址命名了一组应该在这个网络中应用接收到一个分组的站点。多播地址的最高位必须是 1110，范围从 224.0.0.0～239.255.255.255。

5. 特殊的地址

每一个字节都为 0 的地址(0.0.0.0)表示当前主机，IP 地址中的每一个字节都为 1 的 IP 地址(255.255.255.255)是当前子网的广播地址，IP 地址中凡是以 11110 开头的 E 类 IP 地址都保留用于将来和实验使用。

IP 地址中不能以十进制 127 作为开头，而以数字 127.0.0.1～127.255.255.255 段的 IP 地址称为回环地址，用于回路测试，如 127.0.0.1 可以代表本机 IP 地址，网络 ID 的第一个 8 位组也不能全置为 0，全 0 表示本地网络。

3.5.2 子网掩码

子网掩码(subnet mask)又名网络掩码、地址掩码，它是一种用来指明一个 IP 地址的哪些位标识的是主机所在的子网，以及哪些位标识的是主机的位掩码。

通常来讲，子网掩码不能单独存在，它必须结合 IP 地址一起使用。子网掩码只有一个作用，就是将某个 IP 地址划分成网络地址和主机地址两部分。

子网掩码是一个 32 位地址，用于屏蔽 IP 地址的一部分以区别网络标识和主机标识，并说明该 IP 地址是在局域网上，还是在远程网上。

对于 A 类地址，默认的子网掩码是 255.0.0.0，而对于 B 类地址来说默认的子网掩码是 255.255.0.0，对于 C 类地址来说默认的子网掩码是 255.255.255.0。

互联网是由各种小型网络构成的，每个网络上都有许多主机，这样便构成了一个有层次的结构。IP 地址在设计时就考虑到地址分配的层次特点，将每个 IP 地址都分割成网络号和主机号两部分，以便于 IP 地址的寻址操作。

子网掩码的设定必须遵循一定的规则。与二进制 IP 地址相同，子网掩码由 1 和 0 组成，且 1 和 0 分别连续。子网掩码的长度也是 32 位，左边是网络位，用二进制数字 1 表示，1 的数目等于网络位的长度；右边是主机位，用二进制数字 0 表示，0 的数目等于主机位的长度。

3.5.3 网关地址

网关(gateway)是一个网络连接到另一个网络的“关口”，网关实质上是一个网络通向其他网络的 IP 地址。主要用于不同网络间传输数据。

例如电脑设备上网，如果是接入到同一个交换机，在交换机内部传输数据是不需要经过网关的，但是如果两台设备不在一个交换机网络，则需要在本机配置网关，内网主机的数据通过网关，网关把数据转发到其他的网络的网关，直至找到对方的主机网络，然后返回数据。

3.5.4 MAC 地址

媒体访问控制(media access control，MAC)是物理地址、硬件地址，用来定义网络设备的位置。

在 OSI 模型中，第三层网络层负责 IP 地址，第二层数据链路层则负责 MAC 地址。因此一个主机会有一个 MAC 地址，而每个网络位置会有一个专属于它的 IP 地址。

IP 地址工作在 OSI 参考模型的第三层网络层。两者之间分工明确，默契合作，完成通信过程。IP 地址专注于网络层，将数据包从一个网络转发到另外一个网络；而 MAC 地址则专注于数据链路层，将一个数据帧从一个节点传送到相同链路的另一个节点。

IP 地址和 MAC 地址一般是成对出现的。如果一台计算机要和网络中另一台计算机通信，那么这两台设备必须配置 IP 地址和 MAC 地址，而 MAC 地址是网卡出厂时设定的，这样配置的 IP 地址就和 MAC 地址形成了一种对应关系。

在数据通信时，IP 地址负责表示计算机的网络层地址，网络层设备(如路由器)根据 IP 地址来进行操作；MAC 地址负责表示计算机的数据链路层地址，数据链路层设备，根据

MAC 地址来进行操作。IP 地址和 MAC 地址这种映射关系是通过地址解析协议(address resolution protocol,ARP)来实现的。

3.6 Linux 系统配置 IP

Linux 操作系统安装完毕,那接下来如何让 Linux 操作系统能上外网呢? 以下为 Linux 服务器配置 IP 的方法。

Linux 服务器网卡默认配置文件在/etc/sysconfig/network-scripts/下,命名的名称一般为 ifcfg-eth0、ifcfg-eth1,eth0 表示第一块网卡,eth1 表示第二块网卡,以此类推,例如 DELL R720 标配有 4 块千兆网卡,在系统显示的名称依次为 eth0、eth1、eth2、eth3。

修改服务器网卡 IP 地址命令为 vi /etc/sysconfig/network-scripts/ifcfg-eth0 (注 CentOS 7 网卡名为 ifcfg-eno16777736)。vi 编辑网卡配置文件,默认为 DHCP 方式,配置如下:

```
DEVICE = eth0
BOOTPROTO = dhcp
HWADDR = 00:0c:29:52:c7:4e
ONBOOT = yes
TYPE = Ethernet
```

vi 编辑网卡配置文件,修改 BOOTPROTO 为 DHCP 方式,同时添加 IPADDR、NETMASK、GATEWAY 信息如下:

```
DEVICE = eth0
BOOTPROTO = static
HWADDR = 00:0c:29:52:c7:4e
ONBOOT = yes
TYPE = Ethernet
IPADDR = 192.168.1.103
NETMASK = 255.255.255.0
GATEWAY = 192.168.1.1
```

服务器网卡配置文件,详细参数如下:

- DEVICE=eth0:物理设备名。
- ONBOOT=yes:[yes|no](重启网卡是否激活网卡设备)。
- BOOTPROTO=static:[none|static|bootp|dhcp](不使用协议|静态分配|BOOTP 协议|DHCP 协议)。
- TYPE=Ethernet:网卡类型。
- IPADDR=192.168.1.103:IP 地址。
- NETMASK=255.255.255.0:子网掩码。

- GATEWAY＝192.168.1.1：网关地址。

服务器网卡配置完毕后，重启网卡服务/etc/init.d/network restart 即可。然后查看 IP 地址，命令为 ifconfig 或者 ip addr show 查看当前服务器所有网卡的 IP 地址。

CentOS 7 Linux 中，如果没有 ifconfig 命令，可以用 ip addr list/show 查看，也可以安装 ifconfig 命令，需安装软件包 net-tools，命令如下，详细配置如图 3-9 所示。

```
yum install net - tools  - y
```

```
[root@localhost ~]# ifconfig
-bash: ifconfig: command not found
[root@localhost ~]# yum install net-tools -y
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.btte.net
 * extras: mirror.bit.edu.cn
 * updates: mirrors.tuna.tsinghua.edu.cn
Resolving Dependencies
--> Running transaction check
---> Package net-tools.x86_64 0:2.0-0.17.20131004git.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

图 3-9　YUM 安装 net-tools 工具

3.7　Linux 系统配置 DNS

网卡 IP 地址配置完毕后，如果服务器需上外网，还需要配置域名解析地址（domain name system，DNS），DNS 主要用于将请求的域名转换为 IP 地址，DNS 地址配置方法如下：

修改 vi/etc/resolv.conf 文件，加入如下两行代码：

```
nameserver 202.106.0.20
nameserver 8.8.8.8
```

上述语句分别表示主 DNS 与备 DNS，DNS 配置完毕后，无须重启网络服务，DNS 立即生效。用户可以用命令 ping　-c 6　www.baidu.com 查看返回结果，如果有 IP 返回，则表示服务器 DNS 配置正确，如图 3-10 所示。

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ping -c 6 www.baidu.com
PING www.a.shifen.com (103.235.46.39) 56(84) bytes of data.
64 bytes from 103.235.46.39: icmp_seq=1 ttl=47 time=150 ms
64 bytes from 103.235.46.39: icmp_seq=2 ttl=47 time=146 ms
64 bytes from 103.235.46.39: icmp_seq=3 ttl=47 time=145 ms
64 bytes from 103.235.46.39: icmp_seq=4 ttl=47 time=163 ms
64 bytes from 103.235.46.39: icmp_seq=5 ttl=47 time=158 ms
64 bytes from 103.235.46.39: icmp_seq=6 ttl=47 time=165 ms

--- www.a.shifen.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5171ms
rtt min/avg/max/mdev = 145.050/154.883/165.090/7.825 ms
[root@www-jfedu-net ~]#
```

图 3-10　ping 命令返回值

3.8 Linux 网卡名称命名

CentOS 7 服务器，默认网卡名为 ifcfg-eno16777736，如果用户想把网卡名改成 ifcfg-eth0，按如下步骤操作即可。

（1）编辑/etc/sysconfig/grub 文件，命令为 vi /etc/sysconfig/grub，在倒数第二行 quiet 后加入如下代码，详细配置如图 3-11 所示。

```
net.ifnames = 0 biosdevname = 0
```

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet net.ifnames=0 biosdevname=0"
GRUB_DISABLE_RECOVERY="true"
~
~
~
```

图 3-11　网卡配置 ifnames 设置

（2）执行命令 grub2-mkconfig -o /boot/grub2/grub.cfg，生成新的 grub.cfg 文件，命令如下，详细配置如图 3-12 所示。

```
grub2 - mkconfig  - o /boot/grub2/grub.cfg
```

```
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-327.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-327.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-6fa6fd8dd8874932be8afaae5b91800f
Found initrd image: /boot/initramfs-0-rescue-6fa6fd8dd8874932be8afaae5b91800f
done
[root@localhost ~]#
```

图 3-12　生成新的 grub.cnf 文件

（3）重命名网卡名称，执行命令 mv ifcfg-eno16777736 ifcfg-eth0，修改 ifcfg-eth0 文件中 DEVICE= eno16777736 为 DEVICE= eth0，如图 3-13 所示。

```
[root@localhost network-scripts]# ls
ifcfg-eth0    ifdown-isdn      ifup          ifup-plip      ifup-tunn
ifcfg-lo      ifdown-post      ifup-aliases  ifup-plusb     ifup-wire
ifdown        ifdown-ppp       ifup-bnep     ifup-post      init.ipv6
ifdown-bnep   ifdown-routes    ifup-eth      ifup-ppp       network-f
ifdown-eth    ifdown-sit       ifup-ib       ifup-routes    network-f
ifdown-ib     ifdown-Team      ifup-ippp     ifup-sit
ifdown-ippp   ifdown-TeamPort  ifup-ipv6     ifup-Team
ifdown-ipv6   ifdown-tunnel    ifup-isdn     ifup-TeamPort
[root@localhost network-scripts]# mv ifcfg-eno16777736 ifcfg-eth0
```

图 3-13　重命名网卡名称

（4）重启服务器，并验证网卡名称是否为 eth0，Reboot 完成后，如图 3-14 所示。

```
192.168.1.103
Last login: Sat Aug 20 13:41:10 2016 from 192.168.1.101
[root@localhost ~]#
[root@localhost ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 15
        inet 192.168.1.103  netmask 255.255.255.0  broad
        inet6 fe80::20c:29ff:fe1c:9220  prefixlen 64  sc
        ether 00:0c:29:1c:92:20  txqueuelen 1000  (Ether
        RX packets 39  bytes 5322 (5.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 47  bytes 8499 (8.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  co

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
```

图 3-14　验证网卡设备名称

3.9　CentOS 7 密码重置

修改 CentOS 7 root 密码非常简单，只需登录系统，执行命令 passwd 按 Enter 键即可，但是如果忘记 root 密码，无法登录系统，该如何去重置 root 用户的密码呢？以下为重置 root 用户密码的方法。

（1）Reboot 重启系统，系统启动进入欢迎界面，加载内核步骤时，按 E 键，然后选中 CentOS Linux (3.10.0-327.el7.x86_64)7 (Core)，如图 3-15 所示。



图 3-15　内核菜单选择界面

（2）继续按 E 键进入编辑模式，找到 ro crashkernel＝auto xxx 项，将 ro 改成 rw init＝/sysroot/bin/sh，如图 3-16 所示。

（3）修改后如图 3-17 所示。

（4）按 Ctrl＋X 键进入单用户模式，如图 3-18 所示。

```
        insmod part_msdos
        insmod xfs
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hin\
t-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1'  36ff9bf6-e\
8a4-48da-9bba-b9c7a89fe151
        else
          search --no-floppy --fs-uuid --set=root 36ff9bf6-e8a4-48da-9bba-b9c7\
a89fe151
        fi
        linux16 /vmlinuz-3.10.0-327.el7.x86_64 root=UUID=01581e1f-c36c-4968-a5\
b8-7b570e424f1c ro crashkernel=auto rhgb quiet net.ifnames=0 biosdevname=0
        initrd16 /initramfs-3.10.0-327.el7.x86_64.img

      Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
      discard edits and return to the menu. Pressing Tab lists
      possible completions.
```

图 3-16　内核编辑界面

```
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdc
t-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1'  36
8a4-48da-9bba-b9c7a89fe151
        else
          search --no-floppy --fs-uuid --set=root 36ff9bf6-e8a4-48da-9
a89fe151
        fi
        linux16 /vmlinuz-3.10.0-327.el7.x86_64 root=UUID=01581e1f-c36c
b8-7b570e424f1c rw init=/sysroot/bin/sh crashkernel=auto rhgb quiet r
es=0 biosdevname=0
        initrd16 /initramfs-3.10.0-327.el7.x86_64.img

      Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
      discard edits and return to the menu. Pressing Tab lists
      possible completions.
```

图 3-17　内核编辑界面

```
[    2.089608] sd 0:0:0:0: [sda] Assuming drive cache: write through

Generating "/run/initramfs/rdsosreport.txt"
[    3.757949] blk_update_request: I/O error, dev fd0, sector 0
[    4.168323] blk_update_request: I/O error, dev fd0, sector 0

Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot
after mounting them and attach it to a bug report.


:/#
```

图 3-18　进入系统单用户模式

（5）执行命令 chroot /sysroot 访问系统，并使用 passwd 修改 root 密码，如图 3-19 所示。

（6）更新系统信息，touch /. autorelabel，执行命令 touch /. autorelabel，在/目录下创建一个. autorelabel 文件，如果该文件存在，系统在重启时就会对整个文件系统进行 relabeling 重新标记，可以理解为对文件进行底层权限的控制和标记，如果 SELinux 属于 disabled 关闭状态则不需要执行这条命令，如图 3-20 所示。

```
:/# chroot /sysroot
:/#
:/#
:/# passwd root
Changing password for user root.
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
passwd: all authentication tokens updated successfully.
:/# _
```

图 3-19 修改 root 用户密码

```
:/# passwd root
Changing password for user root.
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
passwd: all authentication tokens updated successfully.
:/#
:/# touch /.autorelabel
:/#
:/#
:/#
```

图 3-20 创建 autorelabel 文件

3.10 远程管理 Linux 服务器

Linux 系统安装完毕后，可以通过远程工具来连接到 Linux 服务器，远程连接服务器管理的好处在于可以跨地区管理服务器，例如用户在北京，想管理的服务器在上海某 IDC 机房，通过远程管理，不需要到 IDC 机房现场去操作，直接通过远程工具即可管理服务器。

远程管理 Linux 服务器需要满足以下三个步骤：

(1) 服务器配置 IP 地址，如果服务器在公网，需配置公网 IP，如果服务器在内部局域网，可以直接配置内部私有 IP 即可。

(2) 服务器安装 SSHD 软件服务并启动该服务，几乎所有的 Linux 服务器系统安装完毕均会自动安装并启动 SSHD 服务，SSHD 服务监听 22 端口。

(3) 在服务器中防火墙服务需要允许 22 端口对外开放，初学者可以临时关闭防火墙，CentOS 6 Linux 关闭防火墙的命令为 service iptables stop，而 CentOS 7 Linux 关闭防火墙的命令为 systemctl stop firewalld. service。

常见的 Linux 远程管理工具包括 SecureCRT、Xshell、Putty、Xmanger 等工具。目前主流的远程管理 Linux 服务器工具为 SecureCRT，官网 https://www. vandyke. com 下载并安装 SecureCRT，双击打开，单击左上角 quick connect 快速连接，弹出如图 3-21 所示的界面，连接配置具体步骤如下：

- 协议(P)：选择 SSH2；

- 主机名(H)：输入 Linux 服务器 IP 地址；
- 端口(O)：输入 22；
- 防火墙(F)：选择 None；
- 用户名(U)：输入 root。

单击下方的“连接”按钮，会提示输入密码，输入 root 用户对应密码即可。



图 3-21　SecureCRT 远程 Linux 服务器

通过 SecureCRT 远程连接 Linux 服务器之后，如图 3-22 所示界面，与服务器本地操作界面一样，在命令行可以执行命令，操作结果与在服务器现场操作是一样的。

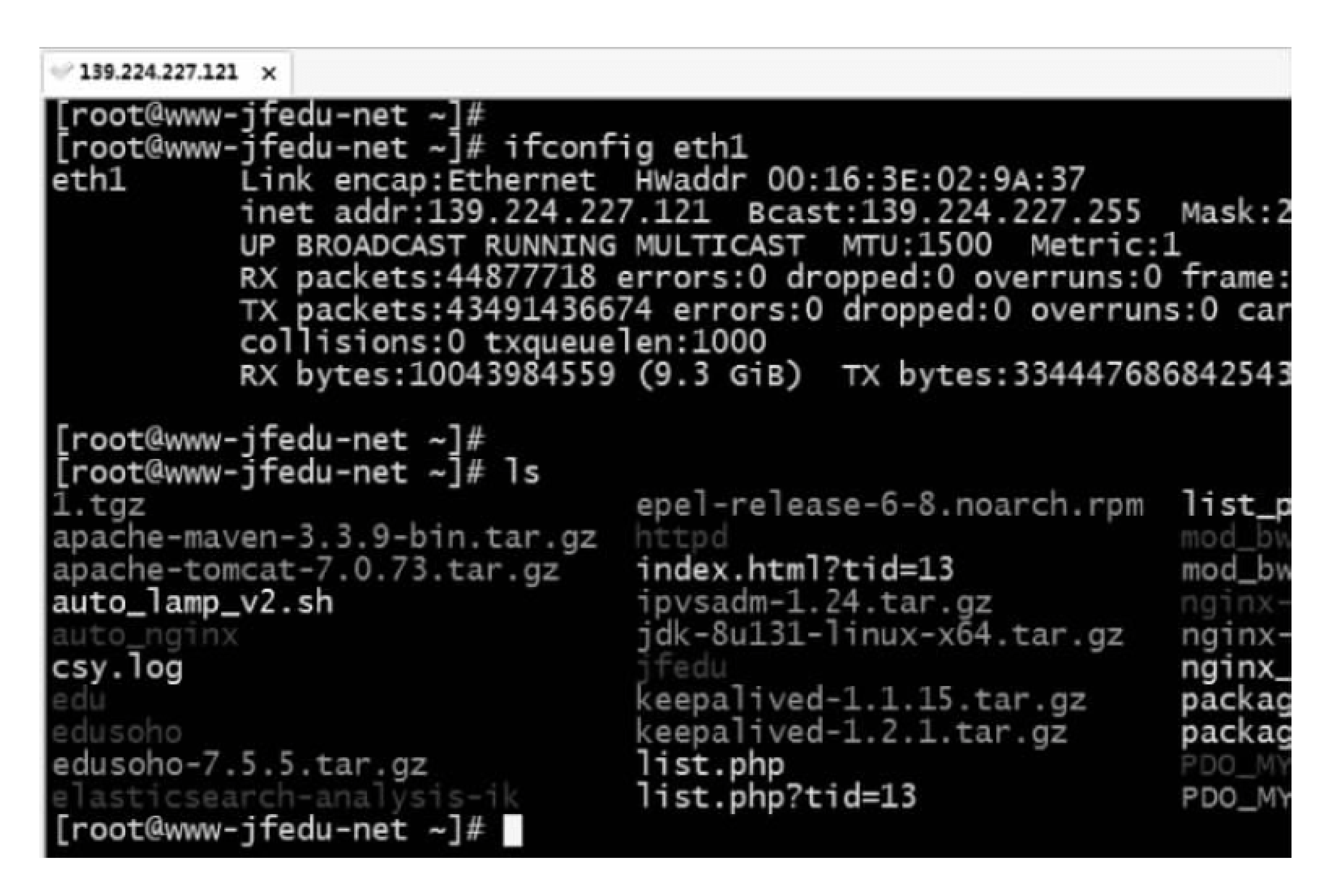


图 3-22　SecureCRT 远程 Linux 服务器

3.11 Linux 系统目录功能

通过以上知识的学习，读者已经能够独立安装和配置 Linux 服务器 IP 并实现远程连接，为了进一步学习 Linux，需熟练掌握 Linux 系统各个目录的功能。

Linux 主要树结构目录包括/、/root、/home、/usr、/bin、/tmp、/sbin、/proc、/boot 等，图 3-23 所示为典型的 Linux 目录结构。

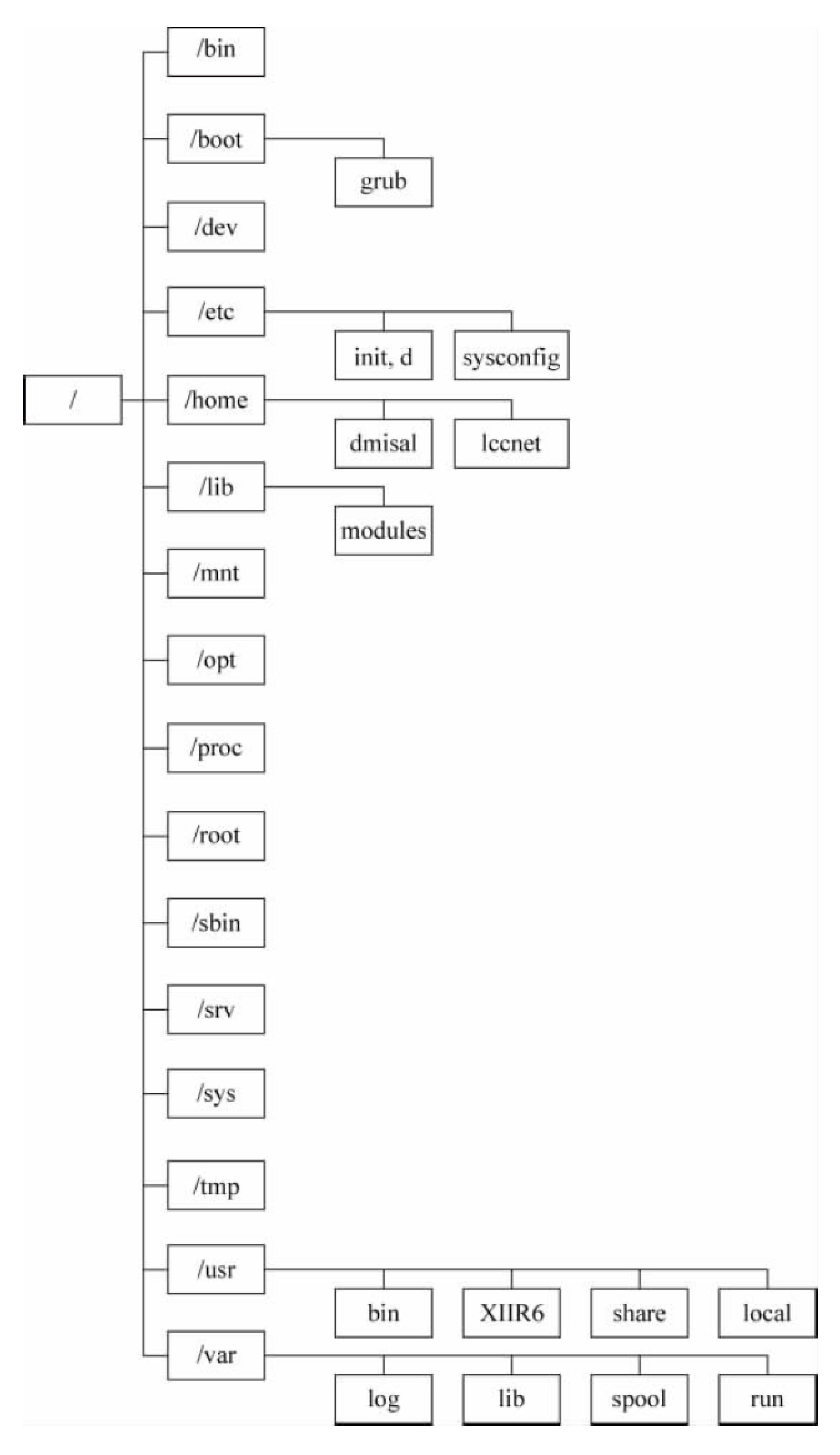


图 3-23 Linux 目录树形结构

Linux 系统中常见目录功能如下：

- /：根目录；
- /bin：存放必要的命令；
- /boot：存放内核以及启动所需的文件；
- /dev：存放硬件设备文件；
- /etc：存放系统配置文件；
- /home：普通用户的宿主目录，用户数据存放在其主目录中；
- /lib|lib64：存放必要的运行库；
- /mnt：存放临时的映射文件系统，通常用来挂载使用；
- /proc：存放存储进程和系统信息；
- /root：超级用户的主目录；
- /sbin：存放系统管理程序；
- /tmp：存放临时文件；
- /usr：存放应用程序，命令程序文件、程序库、手册和其他文档；
- /var：系统默认日志存放目录。

第 4 章



Linux 必备命令

Linux 系统启动默认为字符界面，一般不会启动图形界面，所以应对命令行熟练操作，以便更加高效地管理 Linux 系统。

本章向读者介绍 Linux 系统必备命令各项参数及功能场景，Linux 常见命令包括 cd、ls、pwd、mkdir、rm、cp、mv、touch、cat、head、tail、chmod、chown、echo、df、du、vi/vim、vim 等内容。

4.1　cd 命令详解

cd 命令主要用于目录切换，例如 cd /home 表示切换至/home 目录，cd /root 表示切换至/root 目录，cd ../表示切换至上一级目录，cd ./表示切换至当前目录。其中“.”和“..”可以理解为相对路径，例如 cd ./test 表示以当前目录为参考，表示相对于当前目录，而 cd /home/test 表示完整的路径，理解为绝对路径，如图 4-1 所示。

```
[root@www-jfedu-net tmp]# cd
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# cd /home/
[root@www-jfedu-net home]#
[root@www-jfedu-net home]# cd /root/
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# cd /
[root@www-jfedu-net /]#
[root@www-jfedu-net /]# cd /usr/local/sbin/
[root@www-jfedu-net sbin]#
[root@www-jfedu-net sbin]# cd /root/
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# cd /opt/
[root@www-jfedu-net opt]#
[root@www-jfedu-net opt]#
```

图 4-1　Linux cd 命令操作

4.2　ls 命令详解

ls 命令主要用于浏览目录下的文件或者文件夹，ls ./表示查看当前目录所有的文件和目录，ls -a 表示查看所有的文件，包括隐藏文件、以“.”开头的文件，常用参数详解如下：

- -a，--all：不隐藏任何以“.”开始的项目。
- -A，--almost-all：列出除“.”及“..”以外的任何项目。
- --author：与-l 同时使用时列出每个文件的作者。
- -b，--escape：以八进制溢出序列表示不可打印的字符。
- --block-size=大小：块以指定大小的字节为单位。
- -B，--ignore-backups：不列出任何以“～”字符结束的项目。
- -d，--directory：当遇到目录时列出目录本身而非目录内的文件。
- -D，--dired：产生适合 Emacs 的 dired 模式使用的结果。
- -f：不进行排序，-aU 选项生效，-lst 选项失效。
- -i，--inode：显示每个文件的 inode 号。
- -I，--ignore=PATTERN：不显示任何符合指定 shell PATTERN 的项目。
- -k：即--block-size=1KB。
- -l：使用较长格式列出信息。
- -n，--numeric-uid-gid：类似-l，但列出 UID 及 GID 号。
- -N，--literal：输出未经处理的项目名称(如不特别处理控制字符)。
- -r，--reverse：排序时保留顺序。
- -R，--recursive：递归显示子目录。
- -s，--size：以块数形式显示每个文件分配的尺寸。
- -S：根据文件大小排序。
- -t：根据修改时间排序。
- -u：同-lt 一起使用时按照访问时间排序并显示，同-l 一起使用时显示访问时间并按文件名排序，其他情况则按照访问时间排序。
- -U：不进行排序，按照目录顺序列出项目。
- -v：在文本中进行数字(版本)的自然排序。

4.3 pwd 命令详解

pwd 命令主要用于显示或者查看当前所在的目录路径，如图 4-2 所示。

```
[root@www-jfedu-net ~]# cd
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# pwd
/root
[root@www-jfedu-net ~]# cd /home/
[root@www-jfedu-net home]#
[root@www-jfedu-net home]# pwd
/home
[root@www-jfedu-net home]#
[root@www-jfedu-net home]# cd /tmp/
[root@www-jfedu-net tmp]#
[root@www-jfedu-net tmp]# pwd
/tmp
[root@www-jfedu-net tmp]#
[root@www-jfedu-net tmp]# 
```

图 4-2　pwd 命令查看当前目录

4.4 mkdir 命令详解

mkdir 命令主要用于创建目录，用法为 mkdir dirname，命令后接目录的名称，常用参数详解如下：

用法：mkdir [选项]...目录。若指定目录不存在则创建目录。注意长选项必须使用的参数对于短选项时也是必须使用的。

- -m，--mode=模式：设置权限模式(类似 chmod)，而不是 rwxrwxrwx 减 umask。
- -p，--parents：需要时创建目标目录的上层目录，但即使这些目录已存在也不当作错误处理。
- -v，--verbose：每次创建新目录都显示信息。
- -Z，--context=CTX：将每个创建的目录的 SELinux 安全环境设置为 CTX。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

4.5 rm 命令详解

rm 命令主要用于删除文件或者目录，用法为 rm -rf test.txt (-r 表示递归，-f 表示强制)，常用参数详解如下：

用法：rm [选项]...文件...删除(unlink)文件。

- -f，--force：强制删除，忽略不存在的文件，不提示确认。
- -i：在删除前需要确认。
- -I：在删除超过 3 个文件或者递归删除前要求确认，此选项比-i 提示内容更少，但同样可以阻止大多数错误发生。
- -r，-R，--recursive：递归删除目录及其内容。
- -v，--verbose：详细显示进行的步骤。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

默认时，rm 不会删除目录，使用--recursive(-r 或-R)选项可删除每个给定的目录，以及其下所有的内容。要删除第一个字符为“-”的文件(例如“-foo”)，请使用以下方法之一：

```
rm -- -foo
rm ./-foo
```

4.6 cp 命令详解

cp 命令主要用于复制文件，用法为 cp old.txt /tmp/new.txt，常用来备份，如果复制目录需要加-r 参数，常用参数详解如下：

用法：cp [选项]... [-T]源文件　目标文件

或　cp [选项]...源文件...目录

或　cp [选项]... -t 目录　源文件...

作用为将源文件复制至目标文件，或将多个源文件复制至目标目录。注意长选项必须使用的参数对于短选项时也是必须使用的。

- -a，--archive：等于-dR --preserve=all。
- --backup[=CONTROL：为每个已存在的目标文件创建备份。
- -b：类似--backup，但不接受参数。
- --copy-contents：在递归处理是复制特殊文件内容。
- -d：等于--no-dereference --preserve=links。
- -f，--force：如果目标文件无法打开则将其移除并重试(当-n 选项存在时则不需再选此项)。
- -i，--interactive：覆盖前询问(使前面的-n 选项失效)。
- -H：跟随源文件中的命令行符号链接。
- -l，--link：链接文件而不复制。
- -L，--dereference：总是跟随符号链接。
- -n，--no-clobber：不要覆盖已存在的文件(使前面的-i 选项失效)。
- -P，--no-dereference：不跟随源文件中的符号链接。
- -p：等于--preserve=模式，所有权，时间戳。
- --preserve[=属性列表：保持指定的属性(默认：模式，所有权，时间戳)，如果可能保持附加属性：环境、链接、xattr 等。
- -c：等于 --preserve=context。
- --sno-preserve=属性列表：不保留指定的文件属性。
- --parents：复制前在目标目录创建来源文件路径中的所有目录。
- -R，-r，--recursive：递归复制目录及其子目录内的所有内容。

4.7　mv 命令详解

mv 命令主要用于重命名或者移动文件或者目录，用法为 mv old.txt new.txt，常用参数详解如下：

用法：mv [选项]... [-T]源文件　目标文件

或　mv [选项]...源文件...目录

或　mv [选项]... -t 目录　源文件

作用为将源文件重命名为目标文件，或将源文件移动至指定目录。注意长选项必须使用的参数对于短选项时也是必须使用的。

- --backup[=CONTROL]：为每个已存在的目标文件创建备份。

- -b：类似--backup，但不接受参数。
- -f，--force：覆盖前不询问。
- -i，--interactive：覆盖前询问。
- -n，--no-clobber：不覆盖已存在文件，如果用户指定了-i、-f、-n 中的多个，仅最后一个生效。
- --strip-trailing-slashes：去掉每个源文件参数尾部的斜线。
- -S，--suffix=SUFFIX：替换常用的备份文件后缀。
- -t，--target-directory=DIRECTORY：将所有参数指定的源文件或目录移动至指定目录。
- -T，--no-target-directory：将目标文件视作普通文件处理。
- -u，--update：只在源文件文件比目标文件新或目标文件不存在时才进行移动。
- -v，--verbose：详细显示进行的步骤。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

4.8 touch 命令详解

touch 命令主要用于创建普通文件，用法为 touch test.txt，如果文件存在，则表示修改当前文件时间，常用参数详解如下：

用法：touch [选项]...文件...

作用为将每个文件的访问时间和修改时间改为当前时间。不存在的文件将会被创建为空文件，除非使用-c 或-h 选项。如果文件名为“-”则特殊处理，更改与标准输出相关的文件的访问时间。注意长选项必须使用的参数对于短选项时也是必须使用的。

- -a：只更改访问时间。
- -c，--no-create：不创建任何文件。
- -d，--date=字符串：使用指定字符串表示时间而非当前时间。
- -f：忽略。
- -h，--no-dereference：会影响符号链接本身，而非符号链接所指示的目的地（当系统支持更改符号链接的所有者时，此选项才有用）。
- -m：只更改修改时间。
- -r，--reference=文件：使用指定文件的时间属性而非当前时间。
- -t STAMP：使用[[CC]YY]MMDDhhmm[.ss]格式的时间而非当前时间。
- --time=WORD：使用 WORD 指定的时间。access、atime、use 都等于-a 选项的效果，而 modify、mtime 等于-m 选项的效果。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

4.9 cat 命令详解

cat 命令主要用于查看文件内容，用法为 cat test.txt，可以查看 test.txt 内容，常用参数详解如下：

用法：cat [选项]... [文件]...

作用为将[文件]或标准输入组合输出到标准输出。

- -A，--show-all：等于-vET。
- -b，--number-nonblank：对非空输出行编号。
- -e：等于-vE。
- -E，--show-ends：在每行结束处显示"$"。
- -n，--number：对输出的所有行编号。
- -s，--squeeze-blank：不输出多行空行。
- -t：与-vT 等价。
- -T，--show-tabs：将跳格字符显示为^I。
- -u：被忽略。
- -v，--show-nonprinting：使用^和 M-引用，除了 LFD 和 TAB 之外。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

cat 还有一种用法，即 cat …EOF…EOF，表示追加内容至/tmp/test.txt 文件中，用法如下：

```
cat >>/tmp/test.txt << EOF
My Name is JFEDU.NET
I am From Bei jing.
EOF
```

cat test.txt |more 表示分页显示 text 内容，"|"符号是管道符，用于把"|"前的输出作为后面命令的输入。more 命令常用于分页查看某文件或者内容。

4.10 head 命令详解

head 命令主要用于查看文件内容，通常查看文件前 10 行，head -10 /var/log/messages 可以查看该文件前 10 行的内容，常用参数详解如下：

用法：head [选项]... [文件]...

作用为将每个指定文件的头 10 行显示到标准输出。如果指定了多于一个文件，在每一段输出前会给出文件名作为文件头；如果不指定文件，或者文件为"-"，则从标准输入读取数据。注意长选项必须使用的参数对于短选项时也是必须使用的。

- -q，--quiet，--silent：不显示包含给定文件名的文件头。

- -v，--verbose：总是显示包含给定文件名的文件头。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。
- -c，--bytes=[-]K：显示每个文件的前 K 字节内容，如果附加“-”参数，则除了每个文件的最后 K 字节数据外显示剩余全部内容。
- -n，--lines=[-]K：显示每个文件的前 K 行内容，如果附加“-”参数，则除了每个文件的最后 K 行外显示剩余全部内容。

4.11　tail 命令详解

tail 命令主要用于查看文件内容，通常查看末尾 10 行，用 tail -fn 100 /var/log/messages 可以实时查看该文件末尾 100 行的内容，常用参数详解如下：

用法：tail [选项]… [文件]…

作用为显示每个指定文件的最后 10 行到标准输出。若指定了多于一个文件，程序会在每段输出的开始添加相应文件名作为头。如果不指定文件或文件为“-”，则从标准输入读取数据。注意长选项必须使用的参数对于短选项时也是必须使用的。

- -n，--lines=K：输出的总行数，默认为 10 行。
- -q，--quiet，--silent：不输出给出文件名的头。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。
- -f，--follow[={name|descriptor}]：即时输出文件变化后追加的数据。
- -f，--follow：等于--follow=descriptor。
- -F：即--follow=name -retry。
- -c，--bytes=K：输出最后 K 字节，另外，使用-c +K 从每个文件的第 K 字节输出。

4.12　chmod 命令详解

chmod 命令主要用于修改文件或者目录的权限，例如 chmod o+w　test.txt，赋予 test.txt 其他人 w 写权限，常用参数详解如下：

用法：chmod [选项]…模式[，模式]…文件…

或　　chmod [选项]…八进制模式文件…

或　　chmod [选项]… --reference=参考文件　文件…

作用为将每个文件的模式更改为指定值。

- -c，--changes：类似--verbose，但只在有更改时才显示结果。
- --no-preserve-root：不特殊对待根目录(默认)。
- --preserve-root：禁止对根目录进行递归操作。

- -f，--silent，--quiet：去除大部分的错误信息。
- -R，--recursive：以递归方式更改所有的文件及子目录。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。
- -v，--verbose：为处理的所有文件显示诊断信息。
- --reference=参考文件：使用指定参考文件的模式，而非自行指定权限模式。

4.13 chown 命令详解

chown 命令主要用于文件或者文件夹属主及属组的修改，命令格式例如 chown -R root. root /tmp/test.txt，表示修改 test.txt 文件的用户和组均为 root，常用参数详解如下：

用法：chown [选项]... [所有者][:[组]] 文件...

或 chown [选项]... --reference=参考文件 文件...

作用为更改每个文件的所有者和所属组。当使用--referebce 参数时，将文件的所有者和所属组更改为与指定参考文件相同。

- -f，--silent，--quiet：去除大部分的错误信息。
- --reference=参考文件：使用参考文件的所属组，而非指定值。
- -R，--recursive：递归处理所有的文件及子目录。
- -v，--verbose：为处理的所有文件显示诊断信息。
- -H：命令行参数是一个通到目录的符号链接，则遍历符号链接。
- -L：遍历每一个遇到的通到目录的符号链接。
- -P：遍历任何符号链接(默认)。
- --help：显示帮助信息并退出。
- --version：显示版本信息并退出。

4.14 echo 命令详解

echo 命令主要用于打印字符或者回显，例如输入 echo ok，会显示 ok，echo ok > test.txt 则会把 ok 字符覆盖 test.txt 内容。“>”表示覆盖，原内容被覆盖，“>>”表示追加，原内容不变。例如 echo ok >> test.txt，表示向 test.txt 文件追加 ok 字符，不覆盖原文件里的内容，常用参数详解如下：

使用-e 扩展参数选项时，与如下参数一起使用，有不同含义。

- \a：发出警告声。
- \b：删除前一个字符。
- \c：最后不加上换行符号。
- \f：换行但光标仍旧停留在原来的位置。

- \n：换行且光标移至行首。
- \r：光标移至行首，但不换行。
- \t：插入 tab。
- \v：与\f 相同。
- \\：插入“\”字符。
- echo 打印带颜色字符，常用参数如下：

\033[30m 黑色字 \033[0m
\033[31m 红色字 \033[0m
\033[32m 绿色字 \033[0m
\033[33m 黄色字 \033[0m
\033[34m 蓝色字 \033[0m
\033[35m 紫色字 \033[0m
\033[36m 天蓝字 \033[0m
\033[37m 白色字 \033[0m
\033[40;37m 黑底白字 \033[0m
\033[41;37m 红底白字 \033[0m
\033[42;37m 绿底白字 \033[0m
\033[43;37m 黄底白字 \033[0m
\033[44;37m 蓝底白字 \033[0m
\033[45;37m 紫底白字 \033[0m
\033[46;37m 天蓝底白字 \033[0m
\033[47;30m 白底黑字 \033[0m

echo 颜色打印扩展，auto_lamp_v2.sh 内容如下：

```
echo -e "\033[36mPlease Select Install Menu follow:\033[0m"
echo -e "\033[32m1)Install Apache Server\033[1m"
echo "2)Install MySQL Server"
echo "3)Install PHP Server"
echo "4)Configuration index.php and start LAMP server"
echo -e "\033[31mUsage: { /bin/sh $0 1|2|3|4|help}\033[0m"
```

执行结果如图 4-3 所示。

```
[root@www-jfedu-net ~]# sh auto_lamp_v2.sh
----------------------------

Please Select Install Menu follow:
1)Install Apache Server
2)Install MySQL Server
3)Install PHP Server
4)Configuration index.php and start LAMP server
Usage: { /bin/sh auto_lamp_v2.sh 1|2|3|4|help}
[root@www-jfedu-net ~]#
```

图 4-3　echo -e 颜色打印

4.15 df 命令详解

df 命令常用于磁盘分区查询，常用命令 df -h，查看磁盘分区信息，常用参数详解如下：

用法：df [选项]... [文件]...

作用为显示每个文件所在的文件系统的信息，默认是显示所有文件系统。注意长选项必须使用的参数对于短选项时也是必须使用的。

- -a，--all：显示所有文件系统的使用情况，包括虚拟文件系统。
- -B，--block-size=SIZE：使用字节大小块。
- -h，--human-readable：以人们可读的形式显示大小(例如 1KB、234MB、2GB)。
- -H，--si：同-h，但是强制使用 1000 而不是 1024。
- -i，--inodes：显示 inode 信息而非块使用量。
- -k：即--block-size=1KB。
- -l，--local：只显示本机的文件系统。
- --no-sync：取得使用量数据前不进行同步动作(默认)。
- -P，--portability：使用 POSIX 兼容的输出格式。
- --sync：取得使用量数据前先进行同步动作。
- -t，--type=类型：只显示指定文件系统为指定类型的信息。
- -T，--print-type：显示文件系统类型。
- -x，--exclude-type=类型：只显示文件系统不是指定类型信息。
- --help：显示帮助信息并退出。
- --version：显示版本信息并退出。

4.16 du 命令详解

du 命令常用于查看文件在磁盘中的使用量，常用命令 du -sh，查看当前目录所有文件及文件及的大小，常用参数详解如下：

用法：du [选项]... [文件]...

或　du [选项]... --files0-from=F

作用为计算每个文件的磁盘用量，目录则取总用量。注意长选项必须使用的参数对于短选项时也是必须使用的。

- -a，--all：输出所有文件的磁盘用量，不仅仅是目录。
- --apparent-size：显示表面用量，而并非是磁盘用量，虽然表面用量通常会小一些，但有时它会因为稀疏文件间的“洞”、内部碎片、非直接引用的块等原因而变大。
- -B，--block-size=大小：使用指定字节数的块。
- -b，--bytes：等于--apparent-size --block-size=1。

- -c，--total：显示总计信息。
- -H：等于--dereference-args (-D)。
- -h，--human-readable：以可读性较好的方式显示尺寸(例如 1KB、234MB、2GB)。
- --si：类似-h，但在计算时使用 1000 为基底而非 1024。
- -k：等于--block-size=1KB。
- -l，--count-links：如果是硬链接，就多次计算其尺寸。
- -m：等于--block-size=1MB。
- -L，--dereference：找出任何符号链接指示的真正目的地。
- -P，--no-dereference：不跟随任何符号链接(默认)。
- -0，--null：将每个空行视作 0 字节而非换行符。
- -S，--separate-dirs：不包括子目录的占用量。
- -s，--summarize：只分别计算命令列中每个参数所占的总用量。
- -x，--one-file-system：跳过处于不同文件系统之上的目录。
- -X，--exclude-from=文件：排除与指定文件中描述的模式相符的文件。
- -D，--dereference-args：解除命令行中列出的符号连接。
- --files0-from=F：计算文件 F 中以 NUL 结尾的文件名对应占用的磁盘空间，如果 F 的值是“-”，则从标准输入读入文件名。

以上为 Linux 初学者必备命令，当然 Linux 命令还有很多，后面章节会随时学习新的命令。

4.17 vi/vim 编辑器实战

vi 是一个命令行界面下的文本编辑工具，最早在 1976 年由 Bill Joy 开发，当时名称为 ex。vi 支持绝大多数操作系统(最早在 BSD 上发布)，并且功能已经十分强大。1991 年 Bram Moolenaar 基于 vi 进行改进，发布了 vim，并加入了对 GUI 的支持。

随着 vim 更新发展，vim 已经不是普通意义上的文本编辑器，而是被广泛地应用在文本编辑、方本处理、代码开发等用途，Linux 中主流的文本编辑器包括 vi、vim、sublime、emacs、light table、eclipse、gedit 等。

vim 强大的编辑能力中很大部分是来自于其普通模式命令。vim 的设计理念是命令的组合。例如：

- 5dd：5 表示总共 5 行，删除光标所在后的 5 行，包含光标行。
- d $：“$”代表行尾，删除到行尾的内容，包含光标。
- 2yy：表示复制光标及后 2 行，包括光标行。
- %d：“%”代表全部或者全局，“%d”表示删除文本所有的内容，也即是清空文档所有的内容。

vim 是一个主流开源的编辑器，在 shell 终端执行 vim 命令，会打开编辑器，同时会显示

帮助乌干达贫困的孩子画面，如图 4-4 为 vim 与键盘键位功能对应关系。

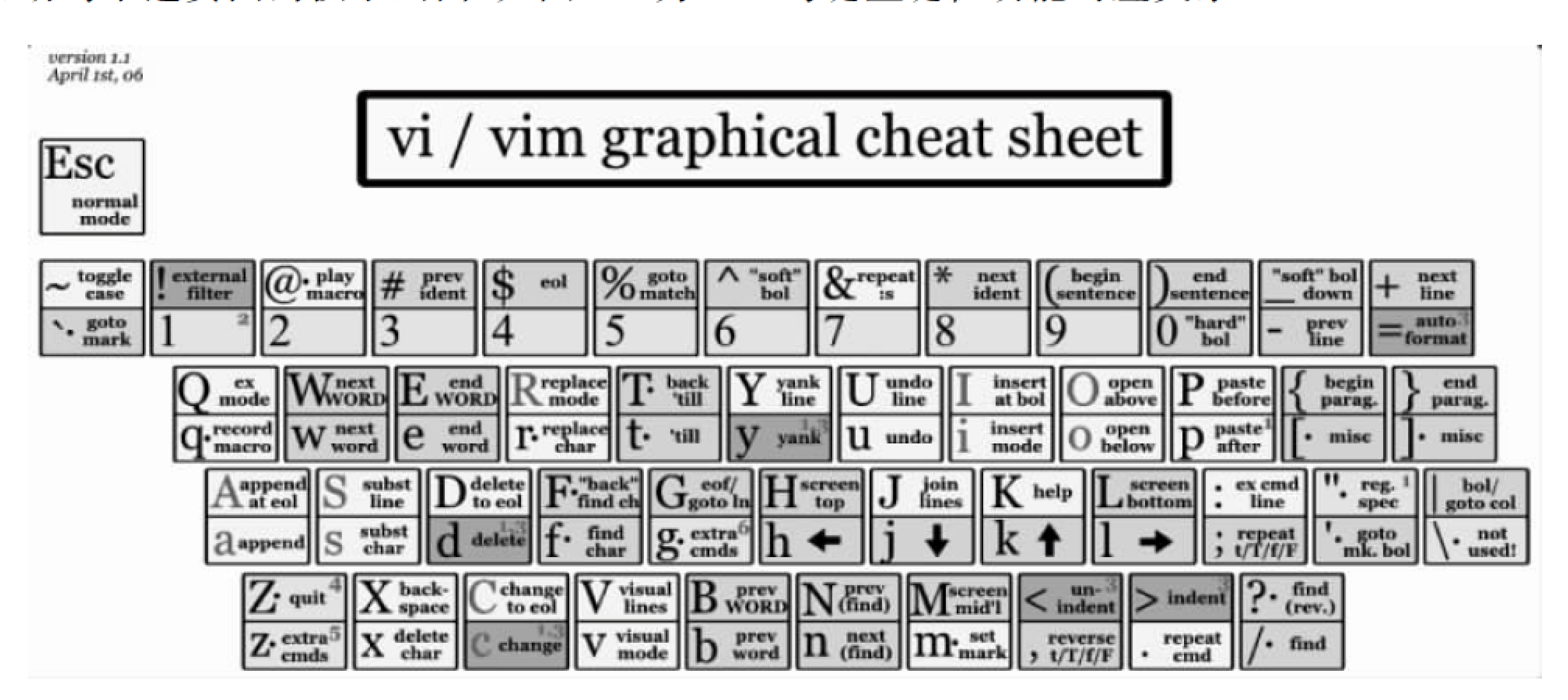


图 4-4 vim 与键盘位置对应关系

4.18 vim 编辑器模式

vim 编辑器模式常用有三种，分别为：

- 命令行模式；
- 文本输入模式；
- 末行模式。

vim 是 vi 的升级版本，它是安装在 Linux 操作系统中的一个软件，官网为 www.vim.org。在 Linux shell 终端下默认执行 vim 命令，按 Enter 键后：

- 默认进入命令行模式；
- 在命令行模式按 I 键进入文本输入模式；
- 按 Esc 键进入命令行模式；
- 按：键进入末行模式。

4.19 vim 编辑器必备

vim 编辑器最强大的功能就在于内部命令及规则使用，以下为 vim 编辑器最常用的语法及规则。

- 命令行模式：可以删除、复制、粘贴、撤销，可以切换到输入模式，输入模式跳转至命令行模式，按 Esc 键。常用命令详解如下：

yy：复制光标所在行。

nyy：复制 n 行。

3yy：复制 3 行。

p,P：粘贴。
yw：复制光标所在的词组，不会复制标点符号。
3yw：复制三个词组。
u：撤销上一次。
U：撤销当前所有。
dd：删除整行。
ndd：删除 n 行。
x：删除一个字符。
u：逐行撤销。
dw：删除一个词组。
a：从光标所在字符后一个位置开始录入。
A：从光标所在行的行尾开始录入。
i：从光标所在字符前一个位置开始录入。
I：从光标所在行的行首开始录入。
o：跳至光标所在行的下一行行首开始录入。
O：跳至光标所在行的上一行行首开始录入。
R：从光标所在位置开始替换。

- 末行模式主要功能包括查找、替换、末行保存、退出等。常用命令详解如下：

:w：保存。
:q：退出。
:s/x/y：替换 1 行。
:wq：保存退出。
1,5sx/y：替换 1,5 行。
:wq!：强制保存退出。
1,$sx/y：从第一行到最后一行。
:q!：强制退出。
:x：保存。
/word：从前往后找，正向搜索。
? word：从后往前找，反向搜索。
:s/old/new/g：将 old 替换为 new，前提是光标一定要移到那一行。
:s/old/new：将这一行中的第一次出现的 old 替换为 new，只替换第一个。
:1,$s/old/new/g：第一行到最后一行中的 old 替换为 new。
:1,2,3s/old/new/g：第一行第二行第三行中的 old 改为 new。
vim +2 jfedu.txt：打开 jfedu.txt 文件，并将光标定位在第二行。
vim +/string jfedu.txt：打开 jfedu.txt 文件，并搜索关键词。

本章小结

通过对本章内容的学习，读者对 Linux 操作系统引导有了进一步的理解，能够快速解决 Linux 启动过程中的故障，同时学习了 CentOS 6 与 CentOS 7 系统的区别，理解了 TCP/IP 协议及 IP 地址相关基础内容。

学会了 Linux 初学者必备的 Linux 命令，能使用命令熟练的操作 Linux 系统，通过对 vim 编辑器的深入学习，能够熟练编辑、修改系统中任意的文本及重要的配置文件。对 Linux 系统的认识及操作有了更进一步的飞跃。

同步作业

1. 修改密码的命令为 passwd，需要按 Enter 键两次，如何一条命令快速修改密码呢？

2. 企业服务器，某天发现系统访问很慢，需要查看系统内核日志，请写出查看系统内核日志的命令。

3. 如何在 Linux 系统/tmp 目录快速创建 1000 个目录，目录为 jfedu1、jfedu2、jfedu3……以此类推，不断增加。

4. httpd. conf 配置文件中存在很多以＃号开头的行，请使用 vim 相关指令删除＃开头的行。

第 5 章



Linux 用户及权限管理

Linux 是一个多用户的操作系统，引入用户，可以更加方便地管理 Linux 服务器。系统默认需要以一个用户的身份登入，而且在系统上启动进程也需要以一个用户身份启动运行，用户可以限制某些进程对特定资源的权限控制。

本章向读者介绍 Linux 系统如何管理、创建、删除、修改用户角色，用户权限配置，组权限配置及特殊权限等内容。

5.1 Linux 用户及组

Linux 操作系统对多用户的管理是非常烦琐的，所以用组的概念来管理用户就变得简单，每个用户可以在一个独立的组，每个组也可以有零个用户或者多个用户。Linux 系统用户是根据用户 ID 来识别的，默认 ID 长度为 32 位，默认 ID 编号从 0 开始，但是为了和老式系统兼容，用户 ID 限制在 60000 以下。Linux 用户总共分为三种，分别如下：

- root 用户 （ID 0）；
- 系统用户 （ID 1～499）；
- 普通用户 （ID 500 以上）。

Linux 系统中的每个文件或者文件夹，都有一个所属用户及所属组，使用 id 命令可以显示当前用户的信息，使用 passwd 命令可以修改当前用户密码。Linux 操作系统用户的特点如下：

- 每个用户拥有一个 UserID，操作系统实际读取的是 UID，而非用户名；
- 每个用户属于一个主组，属于一个或多个附属组，一个用户最多有 31 个附属组；
- 每个组拥有一个 GroupID；
- 每个进程以一个用户身份运行，该用户可对进程拥有资源控制权限；
- 每个可登录用户拥有一个指定的 shell 环境。

5.2 Linux 用户管理

Linux 用户在操作系统中可以进行日常管理和维护，涉及的相关配置文件如下：

- /etc/passwd：保存用户信息。
- /etc/shdaow：保存用户密码(以加密形式保存)。
- /etc/group：保存组信息。
- /etc/login.defs：用户属性、密码过期时间、密码最大长度等限制。
- /etc/default/useradd：显示或更改默认的 useradd 配置文件。

如需创建新用户，可以使用命令 useradd，执行命令 useradd jfedu1 即可创建 jfedu1 用户，同时会创建一个同名的组 jfedu1，默认该用户属于 jfedu1 主组。

useradd jfedu1 命令默认创建用户 jfedu1，会根据如下步骤进行操作：

- 在/etc/passwd 文件中添加用户信息；
- 如使用 passwd 命令创建密码，密码会被加密保存在/etc/shdaow 中；
- 为 jfedu1 创建家目录/home/jfedu1；
- 将/etc/skel 中的.bash 开头的文件复制至/home/jfedu1 家目录；
- 创建与用户名相同的 jfedu1 组，jfedu1 用户默认属于 jfeud1 同名组；
- jfedu1 组信息保存在/etc/group 配置文件中。

在使用 useradd 命令创建用户时，可以支持以下参数：

- -b，--base-dir BASE_DIR：指定新账户的家目录。
- -c，--comment COMMENT：新账户的 GECOS 字段。
- -d，--home-dir HOME_DIR：新账户的主目录。
- -D，--defaults：显示或更改默认的 useradd 配置。
- -e，--expiredate EXPIRE_DATE：新账户的过期日期。
- -f，--inactive INACTIVE：新账户的密码不活动期。
- -g，--gid GROUP：新账户主组的名称或 ID。
- -G，--groups GROUPS：新账户的附加组列表。
- -h，--help：显示此帮助信息并退出。
- -k，--skel SKEL_DIR：使用此目录作为骨架目录。
- -K，--key KEY=VALUE：不使用/etc/login.defs 中的默认值。
- -l，--no-log-init：不要将此用户添加到最近登录和登录失败数据库。
- -m，--create-home：创建用户的主目录。
- -M，--no-create-home：不创建用户的主目录。
- -N，--no-user-group：不创建同名的组。
- -o，--non-unique：允许使用重复的 UID 创建用户。
- -p，--password PASSWORD：加密后的新账户密码。

- -r，--system：创建一个系统账户。
- -R，--root CHROOT_DIR：chroot 到的目录。
- -s，--shell SHELL：新账户的登录 shell。
- -u，--uid UID：新账户的用户 ID。
- -U，--user-group：创建与用户同名的组。
- -Z，--selinux-user SEUSER：为 SELinux 用户映射使用指定 SEUSER。

useradd 案例演示：

（1）新建 jfedu 用户，并加入到 jfedu1，jfedu2 附属组：

```
useradd -G jfedu1,jfedu2 jfedu
```

（2）新建 jfedu3 用户，并指定新的家目录，同时指定其登录的 shell：

```
useradd jfedu3 -d /tmp/ -s /bin/sh
```

5.3 Linux 组管理

所有的 Linux 或者 Windows 系统都有组的概念，通过组可以更加方便地管理用户。组的概念应用于各种行业，例如企业会使用部门、职能或地理区域的分类方式来管理成员，映射在 Linux 系统，同样可以创建用户，并用组的概念对其管理。

Linux 组管理有如下特点：

- 每个组有一个组 ID；
- 组信息保存在/etc/group 中；
- 每个用户至少拥有一个主组，同时还可以拥有 31 个附属组。

通过命令 groupadd、groupdel、groupmod 来对组进行管理，详细参数使用如下。

groupadd 用法：

- -f，--force：如果组已经存在则成功退出，并且如果 GID 已经存在则取消-g。
- -g，--gid GID：为新组使用 GID。
- -h，--help：显示此帮助信息并退出。
- -K，--key KEY=VALUE：不使用/etc/login.defs 中的默认值。
- -o，--non-unique：允许创建有重复 GID 的组。
- -p，--password PASSWORD：为新组使用此加密过的密码。
- -r，--system：创建一个系统账户。

groupmod 用法：

- -g，--gid GID：将组 ID 改为 GID。
- -h，--help：显示此帮助信息并退出。
- -n，--new-name NEW_GROUP：改名为 NEW_GROUP。
- -o，--non-unique：允许使用重复的 GID。

- -p，--password PASSWORD：将密码更改为(加密过的) PASSWORD。

groupdel 用法：

- groupdel jfedu：删除 jfedu 组。

groupadd 案例演示：

(1) groupadd 创建 jingfeng 组：

```
groupadd jingfeng
```

(2) groupadd 创建 jingfeng 组，并指定 GID 为 1000：

```
groupadd -g 1000 jingfeng
```

(3) groupadd 创建一个 system 组，名为 jingfeng 组：

```
groupadd -r jingfeng
```

groupmod 案例演示：

(1) groupmod 修改组名称，将 jingfeng 组名改成 jingfeng1：

```
groupmod -n jingfeng1 jingfeng
```

(2) groupmod 修改组 GID 号，将原 jingfeng1 组 gid 改成 gid 1000：

```
groupmod -g 1000 jingfeng1
```

5.4 Linux 用户及组案例

useradd 主要用于新建用户，而用户新建完毕，可以使用 usermod 来修改用户及组的属性，以下为 usermod 详细参数。

用法：usermod [选项] 登录

常用选项如下：

- -c，--comment：注释 GECOS 字段的新值。
- -d，--home HOME_DIR：用户的新主目录。
- -e，--expiredate EXPIRE_DATE：设定账户过期的日期为 EXPIRE_DATE。
- -f，--inactive INACTIVE：过期 INACTIVE 天数后，设定密码为失效状态。
- -g，--gid GROUP：强制使用 GROUP 为新主组。
- -G，--groups GROUPS：新的附加组列表 GROUPS。
- -a，--append GROUP：将用户追加至上边-G 中提到的附加组中，并不从其他组中删除此用户。
- -h，--help：显示此帮助信息并退出。
- -l，--login LOGIN：新的登录名称。
- -L，--lock：锁定用户账号。

- -m，--move-home：将家目录内容移至新位置(仅与-d 一起使用)。
- -o，--non-unique：允许使用重复的(非唯一的) UID。
- -p，--password PASSWORD：将加密过的密码(PASSWORD)设为新密码。
- -R，--root CHROOT_DIR：chroot 到的目录。
- -s，--shell SHELL：该用户账号的新登录 shell 环境。
- -u，--uid UID：用户账号的新 UID。
- -U，--unlock：解锁用户账号。
- -Z，--selinux-user SEUSER：用户账户的新 SELinux 用户映射。

Usermod 案例演示：

(1) 将 jfedu 用户属组修改为 jfedu1,jfedu2 附属组：

```
usermod -G jfedu1,jfedu2 jfedu
```

(2) 将 jfedu 用户加入到 jfedu3,jfedu4 附属组,-a 为添加新组,原组保留：

```
usermod -a -G jfedu3,jfedu4 jfedu
```

(3) 修改 jfedu 用户,并指定新的家目录,同时指定其登录的 shell：

```
usermod -d /tmp/ -s /bin/sh jfedu
```

(4) 将 jfedu 用户名修改为 jfedu1：

```
usermod -l jfedu1 jfedu
```

(5) 锁定 jfedu1 用户及解锁 jfedu1 用户方法：

```
usermod -L jfedu1; usermod -U jfedu1
```

userdel 案例演示：

使用 userdel 可以删除指定用户及其用户的邮箱目录或者 SELinux 映射环境,详细参数如下：

- userdel jfedu1：保留用户的家目录。
- userdel -r jfedu1：删除用户及用户家目录,用户 login 系统无法删除。
- userdel -rf jfedu1：强制删除用户及该用户家目录,不论是否 login 系统。

5.5 Linux 权限管理

Linux 权限是操作系统用来限制对资源访问的机制,权限一般分为读、写、执行。系统中每个文件都拥有特定的权限、所属用户及所属组,通过这样的机制来限制哪些用户或用户组可以对特定文件进行相应的操作。

Linux 每个进程都是以某个用户身份运行,进程的权限与该用户的权限一样,用户的权限越大,则进程拥有的权限就越大。

Linux 中有的文件及文件夹都有至少三种权限，常见的权限如表 5-1 所示。

表 5-1　Linux 文件及文件夹的权限

权　　限	对文件的影响	对目录的影响
r(读取)	可读取文件内容	可列出目录内容
w(写入)	可修改文件内容	可在目录中创建删除内容
x(执行)	可作为命令执行	可访问目录内容
目录必须拥有 x 权限，否则无法查看其内容		

Linux 权限授权，默认是授权给三种角色，分别是 user、group、other，Linux 权限与用户之间的关联如下：

- U 代表 user，G 代表 group，O 代表 other；
- 每个文件的权限基于 UGO 进行设置；
- 权限三位一组(rwx)，同时需授权给三种角色，即 UGO；
- 每个文件拥有一个所属用户和所属组，对应 UGO，不属于该文件所属用户或所属组使用 O 来表示。

在 Linux 系统中，可以通过 ls -l 查看 jfedu.net 目录的详细属性，目录如下，详细属性如图 5-1 所示。

```
drwxrwxr-x  2  jfedu1 jfedu1 4096  Dec 10 01:36  jfedu.net
```



图 5-1　Linux jfedu.net 目录详细属性

jfedu.net 目录属性参数详解如下：

- d：表示目录，同一位置如果为“-”则表示普通文件。
- rwxrwxr-x：表示三种角色的权限，每三位为一种角色，依次为 U，G，O 权限，如上则表示 user 的权限为 rwx，group 的权限为 rwx，other 的权限为 r-x。
- 2：表示文件夹的链接数量，可理解为该目录下子目录的数量。
- jfedu1：从左到右，第一个 jfedu1 表示该用户名，第二个 jfedu1 则为组名，其他人角

色默认不显示。

- 4096：表示该文件夹占据的字节数。
- Dec 10 01:36：表示文件创建或者修改的时间。
- jfedu.net：表示目录名或者文件名。

5.6　chown 属主及属组

修改某个用户、组对文件夹的属主及属组，用命令 chown 实现，案例演示如下。

(1) 修改 jfedu.net 文件夹所属的用户为 root，其中-R 参数表示递归处理所有的文件及子目录：

```
chown -R root jfedu.net
```

(2) 修改 jfedu.net 文件夹所属的组为 root：

```
chown -R :root jfedu.net 或者 chgrp -R root jfedu.net
```

(3) 修改 jfedu.net 文件夹所属的用户为 root，组也为 root：

```
chown -R root:root jfedu.net
```

5.7　chmod 用户及组权限

修改某个用户、组对文件夹的权限，用命令 chmod 实现，其中以代指 UGO"，""-""＝"代表加入、删除和等于对应权限，具体案例如下：

(1) 授予用户对 jfedu.net 目录拥有 rwx 权限：

```
chmod -R u+rwx jfedu.net
```

(2) 授予组对 jfedu.net 目录拥有 rwx 权限：

```
chmod -R g+rwx jfedu.net
```

(3) 授予用户、组、其他人对 jfedu.net 目录拥有 rwx 权限：

```
chmod -R u+rwx,g+rwx,o+rwx jfedu.net
```

(4) 撤销用户对 jfedu.net 目录拥有 w 权限：

```
chmod -R u-w jfedu.net
```

(5) 撤销用户、组、其他人对 jfedu.net 目录拥有 x 权限：

```
chmod -R u-x,g-x,o-x jfedu.net
```

(6) 授予用户、组、其他人对 jfedu.net 目录只有 rx 权限：

```
chmod -R u=rx,g=rx,o=rx jfedu.net
```

5.8 chmod 二进制权限

Linux 权限默认使用 rwx 来表示，为了更简化在系统中对权限进行配置和修改，Linux 权限引入二进制表示方法，代码如下。

Linux 权限可以将 rwx 用二进制来表示，其中有权限用 1 表示，没有权限用 0 表示。

Linux 权限用二进制显示如下：

```
rwx = 111
r-x = 101
rw-= 110
r--= 100
```

以此类推，转化为十进制，对应十进制结果显示如下：

```
rwx = 111 = 4 + 2 + 1 = 7
r-x = 101 = 4 + 0 + 1 = 5
rw-= 110 = 4 + 4 + 0 = 6
r--= 100 = 4 + 0 + 0 = 4
```

得出结论，用 r=4，w=2，x=1 来表示权限。

使用二进制方式来修改权限案例演示如下(其中默认 jfedu.net 目录权限为 755)：

(1) 授予用户对 jfedu.net 目录拥有 rwx 权限：

```
chmod -R 755 jfedu.net
```

(2) 授予组对 jfedu.net 目录拥有 rwx 权限：

```
chmod -R 775 jfedu.net
```

(3) 授予用户、组、其他人对 jfedu.net 目录拥有 rwx 权限：

```
chmod -R 777 jfedu.net
```

(4) 撤销用户对 jfedu.net 目录拥有 w 权限：

```
chmod -R 555 jfedu.net
```

(5) 撤销用户、组、其他人对 jfedu.net 目录拥有 x 权限：

```
chmod -R 644 jfedu.net
```

(6) 授予用户、组、其他人对 jfedu.net 目录只有 rx 权限：

```
chmod -R 555 jfedu.net
```

5.9 Linux 特殊权限及掩码

Linux 权限除了常见的 rwx 权限之外，还有很多特殊的权限，细心的读者会问，为什么 Linux 目录默认权限为 755，而文件默认权限为 644 呢？这是因为 Linux 权限掩码 umask 导致。

每个 Linux 终端都拥有一个 umask 属性，umaks 属性可以用来确定新建文件、目录的默认权限，默认系统权限掩码为 022。在系统中每创建一个文件或者目录，文件默认权限是 666，而目录权限则为 777，权限对外开放比较大，所以设置了权限掩码之后，默认的文件和目录权限减去 umask 值才是真实的文件和目录的权限。具体说明如下：

- 对应目录权限为 777－022＝755；
- 对应文件权限为 666－022＝644；
- 执行 umask 命令可以查看当前默认的掩码，通过 umask -S 023 可以设置默认的权限掩码。

在 Linux 权限中，除了普通权限外，还有如表 5-2 所示的三个特殊权限。

表 5-2 Linux 三种特殊权限

权限	对文件的影响	对目录的影响
suid	以文件的所属用户身份执行，而非执行文件的用户	无
sgid	以文件所属组身份去执行	在该目录中创建任意新文件的所属组与该目录的所属组相同
sticky	无	对目录拥有写入权限的用户仅可以删除其拥有的文件，无法删除其他用户所拥有的文件

Linux 中设置特殊权限方法如下：

- 设置 suid：chmod u＋s jfedu.net。
- 设置 sgid：chmod g＋s jfedu.net。
- 设置 sticky：chmod o＋t jfedu.net。

特殊权限与设置普通权限一样，可以使用数字方式表示：

- suid＝4；
- sgid＝2；
- sticky＝1。

可以通过 chmod 4755 jfedu.net 对该目录授予特殊权限为 s 的权限，Linux 系统中 s 权限的应用常见包括 su、passwd、sudo，如图 5-2 所示。

```
[root@www-jfedu-net ~]# ll /bin/su
-rwsr-xr-x 1 root root 34904 5月  11 2016 /bin/su
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ll /usr/bin/pa
package-cleanup         pango-querymodules-64    paste
package-stash-conflicts pango-view               patch
pal2rgb                 passwd                   patchperl
[root@www-jfedu-net ~]# ll /usr/bin/passwd
-rwsr-xr-x 1 root root 30768 11月 24 2015 /usr/bin/passwd
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ll /usr/bin/sudo
sudo        sudoedit    sudoreplay
[root@www-jfedu-net ~]# ll /usr/bin/sudo
---s--x--x 1 root root 123832 12月  7 08:36 /usr/bin/sudo
You have new mail in /var/spool/mail/root
[root@www-jfedu-net ~]#
```

图 5-2　Linux 特殊权限 s 应用

本章小结

通过对本章内容的学习，读者可以了解 Linux 用户和组的系统知识，同时掌握 Linux 用户和组在系统中各种案例操作；可以熟练新建用户、删除用户、修改用户属性、添加组、修改组以及删除组。

同步作业

1. 某互联网公司职能及员工信息表如表 5-3 所示，请在 Linux 系统中创建相关员工，并把员工加入到部门。

表 5-3　Linux 用户和组管理

部　　门	职　　能
讲师部(teacher)	jfwu,jfcai
市场部(market)	jfxin,jfqi
管理部(manage)	jfedu,jfteach
运维部(operater)	jfhao,jfyang

2. 批量创建 1～100 个用户，用户名以 jfedu 开头，后面紧跟 1、2、3，例如 jfedu1，jfedu2，jfedu3。

3. 使用 useradd 创建用户并通过-p 参数指定密码，设定完密码需通过系统能正常验证并登录。

4. 小王公司服务器，使用 root 用户通过 SecureCRT 远程登录后，如图 5-3 所示，发现登录终端变成 bash-4.1＃，这是什么原因导致的以及如何修复为正常的登录 shell 环境？请写出答案。

```
bash-4.1# ls
1.tgz                          epel-release-6-8.noarch.rpm  list_python.p
apache-maven-3.3.9-bin.tar.gz  httpd                        mod_bw
apache-tomcat-7.0.73.tar.gz    index.html?tid=13            mod_bw-0.7.tg
auto_lamp_v2.sh                ipvsadm-1.24.tar.gz          nginx-1.6.2
auto_nginx                     jdk-8u131-linux-x64.tar.gz   nginx-1.6.2.t
csy.log                        jfedu                        nginx_instal
edu                            keepalived-1.1.15.tar.gz     package2.xml
edusoho                        keepalived-1.2.1.tar.gz      package.xml
edusoho-7.5.5.tar.gz           list.php                     PDO_MYSQL-1.0
elasticsearch-analysis-ik      list.php?tid=13              PDO_MYSQL-1.0
bash-4.1#
```

图 5-3　SecureCRT 登录 Linux 系统界面

第 6 章



Linux 软件包企业实战

通过前几章的学习，读者掌握了 Linux 系统基本命令、用户及权限等知识。Linux 整个体系的关键不在于系统本身，而是基于 Linux 系统去安装和配置企业中相关的软件、数据及应用程序，所以对软件的维护是运维工程师职责的重中之重。

本章向读者介绍 Linux 系统软件的安装、卸载、配置、维护以及如何构建企业本地 YUM 光盘源及 HTTP 本地源等内容。

6.1 RPM 软件包管理

Linux 软件包从内容上可分为二进制包(binary code)和源码包(source code)，不同类别的软件包使用的管理工具也各不相同。源码包是没有经过编译的包，需要经过 GCC、C++编译器环境编译才能运行，二进制包无须编译，可以直接安装使用。

通常而言，可以通过后缀区别源码包和二进制包，例如以.tar.gz、.zip、.rar 结尾的包称之为源码包，以.rpm 结尾的软件包称之为二进制包。真正区分是否为源码包还是二进制包还得基于软件包里面的文件来判断，例如包含.h、.c、.cpp、.cc 等结尾的源码文件，称之为源码包，而代码里面存在 bin 可执行文件，称之为二进制包。

CentOS 操作系统中有一款默认软件管理的工具，即红帽包管理工具(red hat package manager，RPM)。

使用 RPM 工具可以对软件包实现快速安装、管理及维护。RPM 管理工具适用的操作系统包括 CentOS、Red Hat、Fedora、SUSE 等，RPM 工具常用于管理以.rpm 后缀结尾的软件包。

RPM 包命名格式如下：

```
name-version.rpm
name-version-noarch.rpm
name-version-arch.src.rpm
```

如下软件包格式：

```
epel-release-6-8.noarch.rpm
perl-Pod-Plainer-1.03-1.el6.noarch.rpm
yasm-1.2.0-4.el7.x86_64.rpm
```

RPM 包格式解析如下：

- name：软件名称，例如 yasm、perl-pod-Plainer。
- version：版本号，1.2.0 通用格式为“主版本号.次版本号.修正号”，其中 4 表示发布版本号，意味着该 RPM 包是第几次编译生成的。
- arch：适用的硬件平台，RPM 支持的平台有 i386、i586、i686、x86_64、sparc、alpha 等。
- .rpm：后缀包表示编译好的二进制包，可用 rpm 命令直接安装。
- .src.rpm：源代码包，源码编译生成.rpm 格式的 RPM 包方可使用。
- el*：软件包发行版本，el6 表示该软件包适用于 RHEL 6.X/CentOS 6.X。
- devel：开发包。
- noarch：软件包可以在任何平台上安装。

RPM 工具命令详解如下：

- -a，--all：查询所有已安装软件包。
- -q，--query：表示询问用户，输出信息。
- -l，--list：打印软件包的列表。
- -f，--file：file 查询包含 file 的软件包。
- -i，--info：显示软件包信息，包括名称，版本、描述。
- -v，--verbose：打印输出详细信息。
- -U，--upgrade：升级 RPM 软件包。
- -h，--hash：软件安装，可以打印安装进度条。
- --last：列出软件包时，以安装时间排序，最新的在上面。
- -e，--erase：卸载 RPM 软件包。
- --force：表示强制，强制安装或者卸载。
- --nodeps：RPM 包不依赖。
- -l，--list：列出软件包中的文件。
- --provides：列出软件包提供的特性。
- -R，--requires：列出软件包依赖的其他软件包。
- --scripts：列出软件包自定义的小程序。

RPM 企业案例演示：

- rpm -q httpd：检查 httpd 包是否安装。
- rpm -ql httpd：查看软件安装的路径。
- rpm -qi httpd：查看软件安装的版本信息。

- rpm -e httpd：卸载 httpd 软件。
- rpm -e --nodeps httpd：强制卸载 httpd。
- rpm -qa|grep httpd：检查 httpd 相关的软件是否安装。
- rpm -ivh httpd-2.4.10-el7.x86_64.rpm：安装 httpd 软件。
- rpm -Uvh httpd-2.4.10-el7.x86_64.rpm：升级 httpd 软件。
- rpm -ivh --nodeps httpd-2.4.10-el7.x86_64.rpm：不依赖其他软件包。

6.2 tar 软件包管理

Linux 操作系统除了使用 RPM 管理工具对二进制软件包管理之外，还可以通过 tar、zip、jar 等工具对源码包软件进行管理。

6.2.1 tar 命令参数详解

tar 命令参数详解如下：

- -A，--catenate，--concatenate：将存档与已有的存档合并。
- -c，--create：建立新的存档。
- -d，--diff，--compare：比较存档与当前文件的不同之处。
- --delete：从存档中删除。
- -r，--append：附加到存档结尾。
- -t，--list：列出存档中文件的目录。
- -u，--update：仅将较新的文件附加到存档中。
- -x，--extract，--get：解压文件。
- -j，--bzip2，--bunzip2：有 bz2 属性的软件包。
- -z，--gzip，--ungzip：有 gz 属性的软件包。
- -b，--block-size N：指定块大小为 N * 512 字节(默认时 N=20)。
- -B，--read-full-blocks：读取时重组块。
- -C，--directory DIR：指定新的目录。
- --checkpoint：读取存档时显示目录名。
- -f，--file [HOSTNAME:]F：指定存档或设备，后接文件名称。
- --force-local：强制使用本地存档，即使存在克隆。
- -G，--incremental：建立老 GNU 格式的备份。
- -g，--listed-incremental：建立新 GNU 格式的备份。
- -h，--dereference：不转储动态链接，转储动态链接指向的文件。
- -i，--ignore-zeros：忽略存档中的 0 字节块(通常意味着文件结束)。
- --ignore-failed-read：在不可读文件中作 0 标记后再退出。
- -k，--keep-old-files：保存现有文件，从存档中展开时不进行覆盖。

- -K，--starting-file F：从存档文件 F 开始。
- -l，--one-file-system：在本地文件系统中创建存档。
- -L，--tape-length N：在写入 N * 1024 个字节后暂停，等待更换磁盘。
- -m，--modification-time：当从一个档案中恢复文件时，不使用新的时间标签。
- -M，--multi-volume：建立多卷存档，以便在几个磁盘中存放。
- -O，--to-stdout：将文件展开到标准输出。
- -P，--absolute-paths：不要从文件名中去除"/"。
- -v，--verbose：详细显示处理的文件。
- --version：显示 tar 程序的版本号。
- --exclude：file 不把指定文件包含在内。
- -X，--exclude-from FILE：从指定文件中读入不想包含的文件列表。

6.2.2 tar 企业案例演示

tar 企业案例演示如下：

- tar-cvf jfedu. tar. gz jfedu：打包 jfedu 文件或者目录，打包后名称为 jfedu. tar. gz。
- tar-tf jfedu. tar. gz：查看 jfedu. tar. gz 包中内容。
- tar-rf jfedu. tar. gz jfedu.txt：将 jfedu.txt 文件追加到 jfedu. tar. gz 中。
- tar-xvf jfedu. tar. gz：解压 jfedu. tar. gz 程序包。
- tar-czvf jfedu. tar. gz jfedu：使用 gzip 格式打包并压缩 jfedu 目录。
- tar-cjvf jfedu. tar. bz2 jfedu：使用 bzip2 格式打包并压缩 jfedu 目录。
- tar-czf jfedu. tar. gz * -X list.txt：使用 gzip 格式打包并压当前目录所有文件，排除 list.txt 中记录的文件。
- tar-czf jfedu. tar. gz * --exclude=zabbix-3. 2. 4. tar. gz --exclude=nginx-1. 12. 0. tar. gz：使用 gzip 格式打包并压缩所有文件和目录，排除 zabbix-3. 2. 4. tar. gz 和 nginx-1. 12. 0. tar. gz 软件包。

6.2.3 tar 实现 Linux 操作系统备份

tar 命令工具除了用于日常打包、解压源码包之外，最大的亮点还可以用于 Linux 操作系统文件及目录的备份。使用 tar -g 可以基于 GNU 格式做增量备份，备份原理是检查目录和文件的 atime、mtime、ctime 属性是否被修改。文件及目录时间属性详解如下：

- 文件被访问的时间(access time，atime)；
- 文件内容被改变的时间(modified time，mtime)；
- 文件写入、权限更改的时间(change time，ctime)。

总结：更改文件内容 mtime 和 ctime 都会改变，但 ctime 可以在 mtime 未发生变化时被更改。例如修改文件权限文件 mtime 时间不变而 ctime 时间改变。tar 增量备份案例演示步骤如下：

（1）/root 目录创建 jingfeng 文件夹，同时在 jingfeng 文件夹中新建 jf1.txt、jf2.txt 文件，如图 6-1 所示。

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# mkdir jingfeng
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# cd jingfeng/
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]# pwd
/root/jingfeng
[root@www-jfedu-net jingfeng]# ls
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]# touch jf1.txt jf2.txt
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]# ll
total 0
-rw-r--r-- 1 root root 0 May  3 17:29 jf1.txt
-rw-r--r-- 1 root root 0 May  3 17:29 jf2.txt
[root@www-jfedu-net jingfeng]#
```

图 6-1　创建 jingfeng 目录及文件

（2）使用 tar 命令第一次完整备份 jingfeng 文件夹，-g 指定快照 snapshot 文件，第一次没有该文件则会自动创建，如图 6-2 所示。

```
cd /root/jingfeng/
tar -g /data/backup/snapshot -czvf /data/backup/2017jingfeng.tar.gz
```

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# cd /root/jingfeng/
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]# ls
jf1.txt  jf2.txt
[root@www-jfedu-net jingfeng]# tar  -g  /data/backup/snapshot  -czvf  /data/backu
g.tar.gz  *
jf1.txt
jf2.txt
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]# ll /data/backup/
total 8
-rw-r--r-- 1 root root 126 May  3 17:37 2017jingfeng.tar.gz
-rw-r--r-- 1 root root  36 May  3 17:37 snapshot
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]# cat /data/backup/snapshot
GNU tar-1.23-2
1493804222548381799[root@www-jfedu-net jingfeng]#
```

图 6-2　tar 备份 jingfeng 目录中文件

（3）使用 tar 命令第一次完整备份 jingfeng 文件夹会生成快照文件/data/backup/snapshot，后期增量备份会以 snapshot 文件为参考。在 jingfeng 文件夹中再创建 jf3.txt、jf4.txt 文件，然后再通过 tar 命令增量备份 jingfeng 文件夹所有内容，如图 6-3 所示。

```
cd /root/jingfeng/
touch jf3.txt jf4.txt
tar -g /data/backup/snapshot -czvf /data/backup/2017jingfeng_add1.tar.gz *
```

如图 6-3 所示，增量备份时需-g 指定第一次完整备份的快照 snapshot 文件，同时增量打包的文件名不能与第一次备份后的文件名重名，通过 tar -tf 可以查看打包后的文件内容。

```
[root@www-jfedu-net jingfeng]# touch jf3.txt jf4.txt
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]# ll
total 0
-rw-r--r-- 1 root root 0 May  3 17:29 jf1.txt
-rw-r--r-- 1 root root 0 May  3 17:29 jf2.txt
-rw-r--r-- 1 root root 0 May  3 17:42 jf3.txt
-rw-r--r-- 1 root root 0 May  3 17:42 jf4.txt
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]# tar -g /data/backup/snapshot -czvf /data/backup/20
d1.tar.gz *
jf3.txt
jf4.txt
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]# tar -tf /data/backup/2017jingfeng_add1.tar.gz
jf3.txt
jf4.txt
```

图 6-3 tar 增量备份 jingfeng 目录中文件

6.2.4 shell+tar 实现增量备份

企业中日常备份的数据包括/boot、/etc、/root、/data 等目录。备份的策略为每周一至周六执行增量备份，每周日执行全备份。在企业中备份操作系统数据均使用 shell 脚本完成，此处 auto_backup_system.sh 备份脚本供参考。后面章节会系统讲解 shell 脚本，脚本内容如下：

```
#!/bin/bash
#Automatic Backup Linux System Files
#By Author www.jfedu.net
#Define Variables
SOURCE_DIR=(
    $*
)
TARGET_DIR=/data/backup/
YEAR=`date +%Y`
MONTH=`date +%m`
DAY=`date +%d`
WEEK=`date +%u`
FILES=system_backup.tgz
CODE=$?
if
    [ -z $SOURCE_DIR ]; then
echo -e "Please Enter a File or Directory You Need to Backup:\n------------------------
---------------------\nExample $0 /boot /etc ......"
    exit
fi
#Determine Whether the Target Directory Exists
if
    [ ! -d $TARGET_DIR/$YEAR/$MONTH/$DAY ]; then
    mkdir -p $TARGET_DIR/$YEAR/$MONTH/$DAY
    echo "This $TARGET_DIR Created Successfully!"
fi
#EXEC Full_Backup Function Command
```

```
Full_Backup()
{
if
     [ " $WEEK" -eq "7" ]; then
     rm -rf $TARGET_DIR/snapshot
     cd $TARGET_DIR/$YEAR/$MONTH/$DAY ; tar -g $TARGET_DIR/snapshot -czvf $FILES 'echo
${SOURCE_DIR[@]}'
     [ " $CODE" == "0" ]&&echo -e "------------------------------------------
-\nFull_Backup System Files Backup Successfully!"
fi
}
#Perform incremental BACKUP Function Command
Add_Backup()
{
    cd $TARGET_DIR/$YEAR/$MONTH/$DAY ;
if
     [ -f $TARGET_DIR/$YEAR/$MONTH/$DAY/$FILES ]; then
     read -p " $FILES Already Exists, overwrite confirmation yes or no ? : " SURE
     if [ $SURE == "no" -o $SURE == "n" ]; then
     sleep 1 ; exit 0
     fi
#Add_Backup Files System
     if
          [ $WEEK -ne "7" ]; then
          cd $TARGET_DIR/$YEAR/$MONTH/$DAY ; tar -g $TARGET_DIR/snapshot -czvf $$_$FILES
'echo ${SOURCE_DIR[@]}'
          [ " $CODE" == "0" ]&&echo -e "------------------------------------------
-\nAdd_Backup System Files Backup Successfully!"
    fi
else
    if
        [ $WEEK -ne "7" ]; then
        cd $TARGET_DIR/$YEAR/$MONTH/$DAY ; tar -g $TARGET_DIR/snapshot -czvf $FILES 'echo $
{SOURCE_DIR[@]}'
        [ " $CODE" == "0" ]&&echo -e "------------------------------------------
-\nAdd_Backup System Files Backup Successfully!"
    fi
fi
}
Full_Backup; Add_Backup
```

6.3 zip 软件包管理

Zip 是计算机文件的压缩的算法，原名 deflate(真空)，发明者为菲利普·卡兹(Phil Katz)，他于 1989 年 1 月公布了该格式的资料。zip 软件包命名后缀通常使用 zip。

主流的压缩格式包括 tar、rar、zip、war、gzip、bz2、iso 等。性能上 tar、war、rar 格式比 zip 格式压缩率较高，但压缩时间远远高于 zip。zip 工具可以实现对 zip 包进行管理，也可以将文件和文件夹打包成 zip 包。zip 工具打包常见参数详解如下：

- -f：只更改文件。
- -u：只更改或更新文件。
- -d：从压缩文件删除文件。
- -m：将条目移动到 zipfile（删除 OS 文件）。
- -r：递归到目录。
- -j：junk（不记录）目录名。
- -l：将 LF 转换为 CR LF（－11 CR LF～LF）。
- -1：压缩更快，1～9 压缩更好。
- -q：安静操作，不输出执行的过程。
- -v：verbose 操作/打印版本信息。
- -c：添加一行注释。
- -z：添加 zipfile 注释。
- -o：读取名称使 zip 文件与最新条目一样旧。
- -x：不包括以下名称。
- -F：修复 zipfile（-FF 尝试更难）。
- -D：不要添加目录条目。
- -T：测试 zip 文件完整性。
- -X：eXclude、eXtra 文件属性。
- -e：加密“-”，不要压缩这些后缀。
- -h2：显示更多的帮助。

Zip 企业案例演示：

（1）通过 zip 工具打包 jingfeng 文件夹中所有内容，如图 6-4 所示。

```
zip －rv jingfeng.zip /root/jingfeng/
```

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# zip  -rv  jingfeng.zip /root/jingfeng/
updating: root/jingfeng/        (in=0) (out=0) (stored 0%)
updating: root/jingfeng/jf4.txt (in=0) (out=0) (stored 0%)
updating: root/jingfeng/jf1.txt (in=0) (out=0) (stored 0%)
updating: root/jingfeng/jf2.txt (in=0) (out=0) (stored 0%)
updating: root/jingfeng/jf3.txt (in=0) (out=0) (stored 0%)
total bytes=0, compressed=0 -> 0% savings
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ll jingfeng.zip
-rw-r--r-- 1 root root 858 May  3 18:16 jingfeng.zip
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# zip -T jingfeng
test of jingfeng.zip OK
[root@www-jfedu-net ~]#
```

图 6-4 zip 工具对 jingfeng 目录打包备份

（2）通过 zip 工具打包 jingfeng 文件夹中所有内容并排除部分文件，如图 6-5 所示。

```
zip -rv jingfeng.zip * -x jf1.txt
zip -rv jingfeng.zip * -x jf2.txt -x jf3.txt
```

```
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]# zip -rv jingfeng.zip * -x jf1.txt
excluding jf1.txt
file matches zip file -- skipping
updating: jf2.txt        (in=0) (out=0) (stored 0%)
updating: jf3.txt        (in=0) (out=0) (stored 0%)
total bytes=0, compressed=0 -> 0% savings
[root@www-jfedu-net jingfeng]#
[root@www-jfedu-net jingfeng]# zip -rv jingfeng.zip * -x jf2.txt -x jf3.txt
excluding jf2.txt
excluding jf3.txt
file matches zip file -- skipping
updating: jf1.txt        (in=0) (out=0) (stored 0%)
total bytes=0, compressed=0 -> 0% savings
[root@www-jfedu-net jingfeng]#
```

图 6-5　zip 对 jingfeng 目录打包备份，排除部分文件

（3）通过 zip 工具删除 jingfeng.zip 中 jf3.txt 文件，如图 6-6 所示。

```
zip jingfeng.zip -d jf3.txt
```

```
[root@www-jfedu-net ~]# ll jingfeng.zip
-rw-r--r-- 1 root root 858 May  3 18:16 jingfeng.zip
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# unzip jingfeng.zip
Archive:  jingfeng.zip
   creating: root/jingfeng/
 extracting: root/jingfeng/jf4.txt
 extracting: root/jingfeng/jf1.txt
 extracting: root/jingfeng/jf2.txt
 extracting: root/jingfeng/jf3.txt
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# unzip jingfeng.zip -d /data/backup/
Archive:  jingfeng.zip
   creating: /data/backup/root/jingfeng/
 extracting: /data/backup/root/jingfeng/jf4.txt
 extracting: /data/backup/root/jingfeng/jf1.txt
 extracting: /data/backup/root/jingfeng/jf2.txt
 extracting: /data/backup/root/jingfeng/jf3.txt
```

图 6-6　unzip 对 jingfeng 目录解压

（4）通过 unzip 工具解压 jingfeng.zip 文件，如图 6-6 所示。

```
unzip jingfeng.zip
unzip jingfeng.zip -d /data/backup/
```

注意：可以用-d 指定解压后的目录。

6.4　源码包软件安装

通常使用 RPM 工具管理以.rpm 结尾的二进制包，而标准的以.zip、.tar 结尾的源代码包则不能使用 RPM 工具去安装、卸载及升级。源码包安装有以下三个步骤：

- ./configure：预编译，主要用于检测系统基准环境库是否满足 gcc 环境，生成 makefile 文件。

- make：编译，基于第一步生成的 makefile 文件，进行源代码的编译。
- make install：安装，编译完毕之后将相关的可运行文件安装至系统中。

使用 make 编译时 Linux 操作系统必须有 GCC 编译器，用于编译源码。

源码包安装通常需要./configure、make、make install 三个步骤，某些特殊源码可以只有三步中的其中一个或者两个步骤。

以 CentOS 7 Linux 系统为基准，在其上安装 Nginx 源码包，企业中源码安装软件的详细步骤如下：

（1）Nginx.org 官网下载 Nginx-1.13.0.tar.gz 包：

```
wget http://nginx.org/download/nginx-1.13.0.tar.gz
```

（2）Nginx 源码包解压：

```
tar -xvf nginx-1.13.0.tar.gz
```

（3）进入源码包解压后的目录，执行./configure 指令进行预编译，分号"；"表示连接多个命令：

```
cd nginx-1.13.0; ./configure
```

（4）make 编译：

```
make
```

（5）make install 安装：

```
make install
```

通过以上 5 个步骤，源码包软件安装成功。源码包在编译及安装时，可能会遇到各种错误，需要把错误解决之后，再进行下一步安装即可。后面章节会重点针对企业使用的软件进行案例演练。

6.5　YUM 软件包管理

前端软件包管理器(yellow dog updater modified，YUM)适用于 CentOS、Fedora、Red Hat 及 SUSE 等操作系统，主要用于管理 RPM 包。YUM 工具能够从指定的服务器自动下载 RPM 包并且安装，还可以自动处理依赖性关系。

使用 RPM 工具管理和安装软件时，会发现 RPM 包有依赖，需要逐个手动下载安装，而 YUM 工具的最大便利就是可以自动安装所有依赖的软件包，从而提升效率，节省时间。

6.5.1　YUM 工作原理

学习 YUM 一定要理解 YUM 的工作原理。YUM 正常运行需要依赖两个部分：一是 YUM 源端；二是 YUM 客户端。

YUM 客户端安装的所有 RPM 包都是来自 YUM 服务端，YUM 源端通过 HTTP 或者 FTP 服务器发布。YUM 客户端能够从 YUM 源端下载依赖的 RPM 包是由于在 YUM 源端生成了 RPM 包的基准信息，包括 RPM 包版本号、配置文件、二进制信息、依赖关系等。

YUM 客户端需要安装软件或者搜索软件时，会查找/etc/yum.repos.d 下以.repo 结尾文件。CentOS Linux 默认的.repo 文件名为 CentOS-Base.repo，该文件中配置了 YUM 源端的镜像地址，所以每次安装、升级 RPM 包 YUM 客户端均会查找.repo 文件。

YUM 客户端如果配置了 CentOS 官方 repo 源，客户端操作系统必须能联通外网，满足网络条件才能下载软件并安装。如果没有网络，也可以构建光盘源或者内部 YUM 源。YUM 客户端安装软件，默认会把 YUM 源地址、header 信息、软件包、数据库信息、缓存文件存储在/var/cache/yum 中，每次使用 YUM 工具，YUM 优先通过 cache 查找相关软件包，cache 中不存在，然后再访问外网 YUM 源。

6.5.2 YUM 企业案例演练

YUM 工具的使用简便、快捷、高效，在企业中得到广泛的使用，因此得到众多 IT 运维、程序人员的青睐。要能熟练使用 YUM 工具，需要先掌握 YUM 命令行参数的使用。

YUM 命令工具指南，YUM 格式为

```
YUM [command] [package] -y| -q
```

其中的[options]是可选项。-y 表示安装或者卸载出现 yes 时，自动确认 yes；-q 表示不显示安装的过程。YUM 命令工具的参数详解如下：

- yum install httpd：安装 httpd 软件包。
- yum search：YUM 搜索软件包。
- yum list httpd：显示指定程序包安装情况 httpd。
- yum list：显示所有已安装及可安装的软件包。
- yum remove httpd：删除程序包 httpd。
- yum erase httpd：删除程序包 httpd。
- yum update：内核升级或者软件更新。
- yum update httpd：更新 httpd 软件。
- yum check-update：检查可更新的程序。
- yum info httpd：显示安装包信息 httpd。
- yum provides：列出软件包提供哪些文件。
- yum provides "*/rz"：列出 rz 命令由哪个软件包提供。
- yum grouplist：查询可以用 groupinstall 安装的组名称。
- yum groupinstall "Chinese Support"：安装中文支持。
- yum groupremove "Chinese Support"：删除程序组 Chinese Support。
- yum deplist httpd：查看程序 httpd 依赖情况。

- yum clean packages：清除缓存目录下的软件包。
- yum clean headers：清除缓存目录下的 headers。
- yum clean all：清除缓存目录下的软件包及旧的 headers。

YUM 企业案例实战步骤如下：

（1）执行命令 yum install httpd -y，安装 httpd 服务，如图 6-7 所示。

```
[root@www-jfedu-net ~]# yum install httpd -y
Loaded plugins: fastestmirror, langpacks
Repository epel is listed more than once in the configuration
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.6-45.el7.centos.4 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package          Arch             Version                          Reposi
================================================================================
Installing:
 httpd            x86_64           2.4.6-45.el7.centos.4            update

Transaction Summary
================================================================================
Install  1 Package
```

图 6-7　YUM 安装 httpd 软件

（2）执行命令 yum grouplist，检查 groupinstall 的软件组名，如图 6-8 所示。

```
[root@www-jfedu-net ~]# yum grouplist|more
Loaded plugins: fastestmirror, langpacks
Repository epel is listed more than once in the configuration
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
Loading mirror speeds from cached hostfile
Available Environment Groups:
   Minimal Install
   Compute Node
   Infrastructure Server
   File and Print Server
   MATE Desktop
   Basic Web Server
   Virtualization Host
   Server with GUI
   GNOME Desktop
   KDE Plasma Workspaces
   Development and Creative Workstation
```

图 6-8　YUM grouplist 显示组安装名称

（3）执行命令 yum groupinstall "GNOME Desktop" -y，安装 Linux 图像界面，如图 6-9 所示。

```
[root@www-jfedu-net ~]# yum groupinstall "GNOME Desktop" -y
Loaded plugins: fastestmirror, langpacks
Repository epel is listed more than once in the configuration
There is no installed groups file.
Maybe run: yum groups mark convert (see man yum)
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package ModemManager.x86_64 0:1.6.0-2.el7 will be installed
--> Processing Dependency: libqmi-utils for package: ModemManager-1.6.0-2.
--> Processing Dependency: libmbim-utils for package: ModemManager-1.6.0-2
--> Processing Dependency: libqmi-glib.so.5()(64bit) for package: ModemMan
_64
--> Processing Dependency: libmbim-glib.so.4()(64bit) for package: ModemMa
6_64
---> Package NetworkManager-adsl.x86_64 1:1.4.0-19.el7_3 will be installed
```

图 6-9　GNOME Desktop 图像界面安装

（4）执行命令 yum install httpd php php-devel php-mysql mariadb mariadb-server -y，安装中小企业 lamp 架构环境，如图 6-10 所示。

```
[root@www-jfedu-net ~]# yum install httpd php php-devel php-mysql mariadb m
Loaded plugins: fastestmirror, langpacks
Repository epel is listed more than once in the configuration
Loading mirror speeds from cached hostfile
Package httpd-2.4.6-45.el7.centos.4.x86_64 already installed and latest ver
Package php-5.4.16-42.el7.x86_64 already installed and latest version
Package php-devel-5.4.16-42.el7.x86_64 already installed and latest version
Package php-mysql-5.4.16-42.el7.x86_64 already installed and latest version
Package 1:mariadb-5.5.52-1.el7.x86_64 already installed and latest version
Resolving Dependencies
--> Running transaction check
---> Package mariadb-server.x86_64 1:5.5.52-1.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

图 6-10　lamp 中小企业架构安装

（5）执行命令 yum remove ntpdate -y，卸载 ntpdate 软件包，如图 6-11 所示。

```
[root@www-jfedu-net ~]# yum  remove  ntpdate -y
Loaded plugins: fastestmirror, langpacks
Repository epel is listed more than once in the configuration
Resolving Dependencies
--> Running transaction check
---> Package ntpdate.x86_64 0:4.2.6p5-25.el7.centos.2 will be erased
--> Processing Dependency: ntpdate = 4.2.6p5-25.el7.centos.2 for package:
entos.2.x86_64
--> Running transaction check
---> Package ntp.x86_64 0:4.2.6p5-25.el7.centos.2 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package            Arch              Version                              Repo
```

图 6-11　卸载 ntpdate 软件

（6）执行命令 yum provides rz 或者 yum provides "＊/rz"，查找 rz 命令的提供者，如图 6-12 所示。

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# yum provides rz
Loaded plugins: fastestmirror, langpacks
Repository epel is listed more than once in the configuration
Loading mirror speeds from cached hostfile
lrzsz-0.12.20-36.el7.x86_64 : The lrz and lsz modem communications program
Repo        : os
Matched from:
Filename    : /usr/bin/rz


[root@www-jfedu-net ~]#
```

图 6-12　查找 rz 命令的提供者

（7）执行命令 yum update -y，升级 Linux 所有可更新的软件包或 Linux 内核升级，如图 6-13 所示。

```
[root@www-jfedu-net ~]# yum update -y
Loaded plugins: fastestmirror, langpacks
Repository epel is listed more than once in the configuration
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package NetworkManager.x86_64 1:1.0.6-27.el7 will be updated
---> Package NetworkManager.x86_64 1:1.4.0-19.el7_3 will be an update
---> Package NetworkManager-libnm.x86_64 1:1.0.6-27.el7 will be updated
---> Package NetworkManager-libnm.x86_64 1:1.4.0-19.el7_3 will be an updat
---> Package NetworkManager-team.x86_64 1:1.0.6-27.el7 will be updated
---> Package NetworkManager-team.x86_64 1:1.4.0-19.el7_3 will be an update
---> Package NetworkManager-tui.x86_64 1:1.0.6-27.el7 will be updated
---> Package NetworkManager-tui.x86_64 1:1.4.0-19.el7_3 will be an update
---> Package abrt-libs.x86_64 0:2.1.11-36.el7.centos will be updated
---> Package abrt-libs.x86_64 0:2.1.11-45.el7.centos will be an update
```

图 6-13 软件包升级或内核升级

6.6 基于 ISO 镜像构建 YUM 本地源

通常而言，YUM 客户端使用前提是必须联外网，YUM 安装软件会检查.repo 配置文件查找相应的 YUM 源仓库。企业 IDC 机房很多服务器为了安全起见，会禁止服务器上外网，因此不能使用默认的官方 YUM 源仓库，需要自建本地 YUM 源。

构建本地 YUM 光盘源，其原理是通过查找光盘中的软件包实现 YUM 安装软件，配置步骤如下：

（1）将 CentOS-7-x86_64-DVD-1511.iso 镜像加载至虚拟机 CD/DVD 或者放入服务器 CD/DVD 光驱中，并将镜像文件挂载至服务器/mnt 目录，如图 6-14 所示，挂载命令如下：

```
mount /dev/cdrom /mnt/
```

```
[root@www-jfedu-net ~]# cd
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# mount      /dev/cdrom    /mnt/
mount: /dev/sr0 is write-protected, mounting read-only
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# cd /mnt/
[root@www-jfedu-net mnt]#
[root@www-jfedu-net mnt]# ls
CentOS_BuildTag  EULA  images    LiveOS    repodata              RPM-GPG-
EFI              GPL   isolinux  Packages  RPM-GPG-KEY-CentOS-7  TRANS.TB
[root@www-jfedu-net mnt]#
[root@www-jfedu-net mnt]#
```

图 6-14 CentOS ISO 镜像文件挂载

（2）备份/etc/yum.repos.d/CentOS-Base.repo 文件为 CentOS-Base.repo.bak，同时在/etc/yum.repos.d 目录下创建 media.repo 文件，并写入如下内容：

```
[yum]
name = CentOS7
baseurl = file:///mnt
enabled = 1
gpgcheck = 1
```

```
gpgkey = file:///mnt/RPM - GPG - KEY - CentOS - 7
```

media.repo 配置文件详解如下：

- name=CentOS7：YUM 源显示名称。
- baseurl=file:///mnt：ISO 镜像挂载目录。
- gpgcheck=1：是否检查 GPG-KEY。
- enabled=1：是否启用 YUM 源。
- gpgkey=file:///mnt/RPM-GPG-KEY-CentOS-7：指定载目录下的 GPG-KEY 文件验证。

(3) 运行命令 yum clean all 清空 YUM cache，执行 yum install screen -y 安装 screen 软件如图 6-15 所示。

```
[root@www-jfedu-net ~]# yum clean all
Loaded plugins: fastestmirror
Cleaning repos: yum
Cleaning up everything
Cleaning up list of fastest mirrors
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# yum install screen -y
Loaded plugins: fastestmirror
yum
(1/2): yum/group_gz
(2/2): yum/primary_db
Determining fastest mirrors
Resolving Dependencies
--> Running transaction check
---> Package screen.x86_64 0:4.1.0-0.21.20120314git3c2946.el7 will be inst
--> Finished Dependency Resolution

Dependencies Resolved
```

图 6-15 YUM 安装 screen 软件

(4) YUM 光盘源构建完毕，在使用 YUM 源时，会遇到部分软件无法安装，原因是光盘中软件包不完整导致，同时光盘源只能本机使用，其他局域网服务器无法使用。

6.7 基于 HTTP 构建 YUM 网络源

YUM 光盘源默认只能本机使用，局域网其他服务器无法使用 YUM 光盘源，如果想使用的话，需要在每台服务器上构建 YUM 本地源，该方案在企业中不可取，所以需要构建 HTTP 局域网 YUM 源解决。可以通过 createrepo 创建本地 YUM 源端，repo 即为 repository。

构建 HTTP 局域网 YUM 源方法及步骤如下：

(1) 挂载光盘镜像文件至/mnt。

```
mount /dev/cdrom /mnt/
```

(2) 复制/mnt/Packages 目录下所有软件包至/var/www/html/centos/。

```
mkdir - p /var/www/html/centos/
cp - R /mnt/Packages/ * /var/www/html/centos/
```

（3）使用 createrepo 创建本地源，执行如下命令会在 CentOS 目录生成 repodata 目录，目录内容如图 6-16 所示。

```
yum install createrepo* -y
cd /var/www/html
createrepo centos/
```

```
[root@www-jfedu-net html]# pwd
/var/www/html
[root@www-jfedu-net html]#
[root@www-jfedu-net html]# cd centos/
[root@www-jfedu-net centos]#
[root@www-jfedu-net centos]#
[root@www-jfedu-net centos]# cd repodata/
[root@www-jfedu-net repodata]#
[root@www-jfedu-net repodata]# ls
40bac61f2a462557e757c2183511f57d07fba2c0dd63f99b48f0b466b7f2b8d2-other.xml.gz
4cdf96c7618bdbb2dfa543c29496faf76d940f0c04f316c8a3476426c65c81cf-primary.sqlite.bz2
9710c85f1049b4c60c74ae5fd51d3e98e4ecd50a43ab53ff641690fb164a6d63-other.sqlite.bz2
cfa741341d5d270d5b42d6220e2908d053c39a2d8346986bf48cee360e6f7ce8-filelists.xml.gz
d216d0fba4167a2d086c80ac8c708670a7e45ee3295f27ac2702784fc56623b4-primary.xml.gz
d863fcc08a4e8d47382001c3f22693ed77e03815a76cedf34d8256d4c12f6f0d-filelists.sqlite.bz2
repomd.xml
```

图 6-16　createrepo 生成 repodata 目录

（4）利用 HTTP 发布 YUM 本地源。

本地 YUM 源通过 createrepo 搭建完毕，需要借助 HTTP Web 服务器发布/var/www/html/centos/中所有软件，YUM 或者 RPM 安装 HTTP Web 服务器，并启动 httpd 服务。详细步骤如下：

- yum install httpd httpd-devel -y：安装 HTTP Web 服务。
- useradd apache -g apache：创建 Apache 用户和组。
- systemctl restart httpd.service：重启 httpd 服务。
- setenforce 0：临时关闭 SELinux 应用级安全策略。
- systemctl stop firewalld.service：停止防火墙。
- ps -ef |grep httpd：查看 httpd 进程是否启动。

（5）在 YUM 客户端，创建/etc/yum.repos.d/http.repo 文件，输入如下内容：

```
[base]
name = "CentOS7 HTTP YUM"
baseurl = http://192.168.1.115/centos/
gpgcheck = 0
enabled = 1
[updates]
name = "CentOS7 HTTP YUM"
baseurl = http://192.168.1.115/centos
gpgcheck = 0
enabled = 1
```

（6）在 YUM 客户端上执行以下命令，详解如下，结果如图 6-17 所示。

- yum clean all：清空 YUM cache。
- yum install ntpdate -y：安装 ntpdate 软件。

```
[root@www-jfedu-net ~]# yum clean all
Loaded plugins: fastestmirror
Cleaning repos: base updates
Cleaning up everything
Cleaning up list of fastest mirrors
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# yum install ntpdate -y
Loaded plugins: fastestmirror
base
updates
(1/2): base/primary_db
(2/2): updates/primary_db
Determining fastest mirrors
Resolving Dependencies
--> Running transaction check
```

图 6-17　HTTP YUM 源客户端验证

6.8　YUM 源端软件包扩展

默认使用 ISO 镜像构建的 HTTP YUM 源，会发现缺少很多软件包。如果服务器需要挂载移动硬盘，mount 挂载移动硬盘需要 ntfs-3g 软件包支持，而本地光盘镜像中没有该软件包，此时需要往 YUM 源端添加 ntfs-3g 软件包，添加方法如下：

（1）切换至/var/www/html/centos 目录，官网下载 ntfs-3g 软件包。

```
cd /var/www/html/centos/
wget http://dl. fedoraproject. org/pub/epel/7/x86_64/n/ntfs - 3g - 2016. 2. 22 - 3. el7. x86_
64. rpm
http://dl. fedoraproject. org/pub/epel/7/x86_64/n/ntfs - 3g - devel - 2016. 2. 22 - 3. el7. x86_
64. rpm
```

（2）createrepo 命令更新软件包，如需新增其他软件包，把软件下载至本地，然后通过 createrepo 更新即可，如图 6-18 所示。

```
createrepo  -- update centos/
```

```
[root@www-jfedu-net html]#
[root@www-jfedu-net html]# ls centos/ntfs-3g-*
centos/ntfs-3g-2016.2.22-3.el7.x86_64.rpm  centos/ntfs-3g-devel-2016.
[root@www-jfedu-net html]#
[root@www-jfedu-net html]#
[root@www-jfedu-net html]# createrepo --update centos/
Spawning worker 0 with 2 pkgs
Workers Finished
Saving Primary metadata
Saving file lists metadata
Saving other metadata
Generating sqlite DBs
Sqlite DBs complete
```

图 6-18　createrepo update 更新软件包

（3）客户端 YUM 验证，安装 ntfs-3g 软件包，如图 6-19 所示。

```
[root@www-jfedu-net ~]# yum clean all
Loaded plugins: fastestmirror
Cleaning repos: base updates
Cleaning up everything
Cleaning up list of fastest mirrors
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# yum install ntfs-3g -y
Loaded plugins: fastestmirror
base
updates
(1/2): base/primary_db
(2/2): updates/primary_db
Determining fastest mirrors
Resolving Dependencies
--> Running transaction check
---> Package ntfs-3g.x86_64 2:2016.2.22-3.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

图 6-19　YUM install ntfs-3g 软件包

6.9　同步外网 YUM 源

在企业实际应用场景中，仅仅靠光盘中的 RPM 软件包是不能满足需要的，用户可以把外网的 YUM 源中的所有软件包同步至本地，完善本地 YUM 源的软件包数量及完整性。

获取外网 YUM 源软件常见方法包括：rsync、wget、reposync，三种同步方法的区别为：rsync 方式需要外网 YUM 源支持 rsync 协议；wget 可以直接获取；reposync 可以同步几乎所有的 YUM 源。下面以 reporsync 为案例，同步外网 YUM 源软件至本地，步骤如下：

（1）下载 CentOS 7 repo 文件至/etc/yum.repos.d/，并安装 reposync 命令工具。

```
wget http://mirrors.163.com/.help/CentOS7 - Base - 163.repo
mv CentOS7 - Base - 163.repo /etc/yum.repos.d/centos.repo
yum clean all
yum install yum - utils createrepo  - y
yum repolist
```

（2）通过 reposync 命令工具获取外网 YUM 源所有软件包，-r 指定 repolist id，默认不加-r 表示获取外网所有 YUM 软件包，-p 参数表示指定下载软件的路径，如图 6-20 所示。

```
reposync  - r base  - p /var/www/html/centos/
reposync  - r updates  - p /var/www/html/centos/
```

（3）通过 reposync 工具下载完所有的软件包之后，需要执行 createrepo 更新本地 YUM 仓库。

```
createrepo /var/www/html/centos/
```

```
[root@www-jfedu-net ~]# reposync  -r  base    -p    /var/www/html/centos/
No Presto metadata available for base
(1/9299): ORBit2-devel-2.14.19-13.el7.x86_64.rpm
(2/9299): ORBit2-2.14.19-13.el7.i686.rpm
(3/9299): OpenEXR-devel-1.7.1-7.el7.i686.rpm
(4/9299): OpenEXR-devel-1.7.1-7.el7.x86_64.rpm
(5/9299): OpenEXR-1.7.1-7.el7.x86_64.rpm
(6/9299): OpenEXR-libs-1.7.1-7.el7.x86_64.rpm
(7/9299): OpenIPMI-2.0.19-15.el7.x86_64.rpm
(8/9299): OpenEXR-libs-1.7.1-7.el7.i686.rpm
(9/9299): OpenIPMI-devel-2.0.19-15.el7.x86_64.rpm
(10/9299): OpenIPMI-devel-2.0.19-15.el7.i686.rpm
(11/9299): OpenIPMI-libs-2.0.19-15.el7.x86_64.rpm
```

(a) reposync获取外网YUM源软件包(1)

```
[root@www-jfedu-net ~]# reposync  -r  updates    -p    /var/www/html/centos/
(1/1562): 389-ds-base-libs-1.3.5.10-15.el7_3.x86_64.rpm
(2/1562): 389-ds-base-snmp-1.3.5.10-12.el7_3.x86_64.rpm
(3/1562): 389-ds-base-libs-1.3.5.10-20.el7_3.x86_64.rpm
(4/1562): 389-ds-base-snmp-1.3.5.10-15.el7_3.x86_64.rpm
(5/1562): 389-ds-base-snmp-1.3.5.10-18.el7_3.x86_64.rpm
(6/1562): NetworkManager-1.4.0-13.el7_3.x86_64.rpm
(7/1562): 389-ds-base-snmp-1.3.5.10-20.el7_3.x86_64.rpm
(8/1562): NetworkManager-1.4.0-17.el7_3.x86_64.rpm
(9/1562): NetworkManager-1.4.0-19.el7_3.x86_64.rpm
(10/1562): NetworkManager-1.4.0-14.el7_3.x86_64.rpm
(11/1562): NetworkManager-adsl-1.4.0-14.el7_3.x86_64.rpm
(12/1562): NetworkManager-adsl-1.4.0-13.el7_3.x86_64.rpm
(13/1562): NetworkManager-adsl-1.4.0-17.el7_3.x86_64.rpm
(14/1562): NetworkManager-adsl-1.4.0-19.e 0% [                    ]
```

(b) reposync获取外网YUM源软件包(2)

图 6-20　reposync 获取外网 YUM 源软件包

本章小结

通过对本章内容的学习，读者掌握了 Linux 安装不同软件包的工具及命令。使用 RPM 及 YUM 管理以.rpm 结尾的二进制包，基于 configure、make、make install 实现源码包安装，并能够对软件进行安装、卸载及维护。能够独立构建企业光盘源、HTTP YUM 源，实现无外网网络使用 YUM 安装各种软件包及工具，同时能随时添加新的软件包至本地 YUM 源中。

同步作业

1. RPM 及 YUM 管理工具的区别是什么？
2. 企业中安装软件，何时选择 YUM 安装或者源码编译安装？
3. 将 Linux 系统中 PHP5.3 版本升级至 PHP5.5 版本，升级方法有几种，分别写出升级步骤。
4. 使用源码编译安装 httpd-2.4.25.tar.bz2，写出安装的流程及注意事项。
5. 如何将 CentOS 7 Linux 字符界面升级为图形界面，并设置系统启动默认为图形界面。

第 7 章 Linux 磁盘管理



Linux 系统一切以文件的方式存储于硬盘，应用程序数据需要时刻读写硬盘，所以企业生产环境中对硬盘的操作变得尤为重要，对硬盘的维护和管理也是每个运维工程师必做的工作之一。

本章向读者介绍硬盘简介、硬盘数据存储方式、如何在企业生产服务器添加硬盘、对硬盘进行分区、初始化以及对硬盘进行故障修复等内容。

7.1 计算机硬盘简介

硬盘是计算机主要存储媒介之一，由一个或者多个铝制或者玻璃制的碟片组成，碟片外覆盖有铁磁性材料，硬盘内部由磁道、柱面、扇区、磁头等部件组成，如图 7-1 所示。

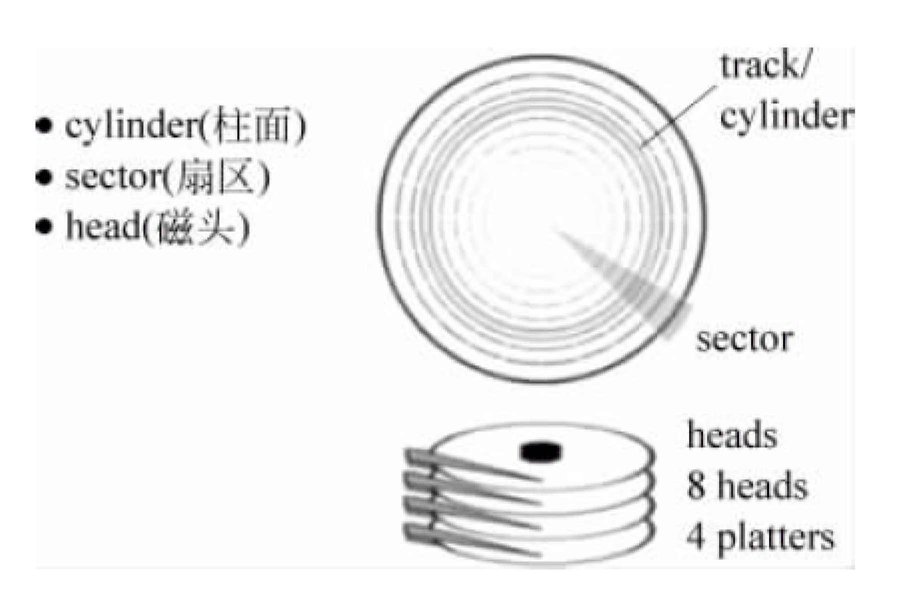


图 7-1 硬盘内部结构组成

Linux 系统中硬件设备相关配置文件存放在/dev/下，常见硬盘命名为/dev/hda、/dev/sda、/dev/sdb、/dev/sdc、/dev/vda。不同硬盘接口，在系统中识别的设备名称不一样。

IDE 硬盘接口在 Linux 中设备名为/dev/hda，SAS、SCSI、SATA 硬盘接口在 Linux 中设备名为 sda，高效云盘硬盘接口会识别为/dev/vda 等。

文件储存在硬盘上，硬盘的最小存储单位叫作 sector(扇区)，每个 sector 储存 512 字节。操作系统在读取硬盘的时候，不会逐个 sector 地去读取，这样效率非常低，为了提升读取效率，操作系统会一次性连续读取多个 sector，即一次性读取多个 sector 称为一个 block(块)。

由多个 sector 组成的 block 是文件存取的最小单位。block 的大小常见的有 1KB、2KB、4KB，block 在 Linux 中常设置为 4KB，即连续 8 个 sector 组成一个 block。

/boot 分区的 block 一般为 1KB，而/data/分区或者/分区的 block 为 4KB。可以通过以下 3 种方法查看 Linux 分区的 block 大小：

```
dumpe2fs /dev/sda1|grep"Block size"
tune2fs -l /dev/sda1|grep"Block size"
stat /boot/|grep"IO Block"
```

例如创建一个普通文件，文件大小为 10B，而默认设置 block 为 4KB，如果有 1 万个小文件，由于每个 block 只能存放一个文件，如果文件的大小比 block 大，会申请更多的 block，相反如果文件的大小比默认 block 小，仍会占用一个 block，这样剩余的空间会被浪费掉。说明如下。

- 1 万个文件理论只占用空间大小：10000 * 10=100000B=97.65625MB。
- 1 万个文件真实占用空间大小：10000 * 4096B=40960000B=40000MB=40GB。
- 根据企业实际需求，此时可以将 block 设置为 1KB，从而节省更多的空间。

7.2 硬盘 block 及 inode 详解

通常而言，操作系统对于文件数据的存放包括两个部分：一是文件内容；二是权限及文件属性。操作系统文件存放是基于文件系统，文件系统会将文件的实际内容存储到 block 中，而将权限与属性等信息存放至 inode 中。

在硬盘分区中，还有一个超级区块（superblock），superblock 会记录整个文件系统的整体信息，包括 inode、block 的总量、使用大小、剩余大小等信息。每个 inode 与 block 都有编号对应，方便 Linux 系统快速定位查找文件。详细说明如下：

- superblock：记录文件系统的整体信息，包括 inode 与 block 的总量、使用大小、剩余大小以及文件系统的格式与相关信息等。
- inode：记录文件的属性、权限，同时会记录该文件的数据所在的 block 编号。
- block：存储文件的内容，如果文件超过默认 block 大小，会自动占用多个 block。

每个 inode 与 block 都有编号，而每个文件都会占用一个 inode，inode 内则有文件数据放置的 block 号码。如果能够找到文件的 inode，就可以找到该文件所放置数据的 block 号码，从而读取该文件内容。

操作系统进行格式化分区时，操作系统自动将硬盘分成两个区域。一个是数据 block 区，用于存放文件数据；另一个是 inode table 区，用于存放 inode 包含的元信息。

每个 inode 节点的大小，可以在格式化时指定，默认为 128B 或 256B，/boot 分区 inode 默认为 128B，其他分区默认为 256B，查看 Linux 系统 inode 的方法如下：

```
dumpe2fs /dev/sda1|grep" Inode size "
tune2fs -l /dev/sda1|grep" Inode size"
stat /boot/|grep"Inode"
```

格式化磁盘时，可以指定默认 inode 和 block 的大小，-b 指定默认 block 值，-I 指定默认 inode 值，如图 7-2 所示，命令如下：

mkfs.ext4 －b 4096 －I 256 /dev/sdb

```
[root@www-jfedu-net ~]# mkfs.ext4 -b 4096 -I 256 /dev/sdb
mke2fs 1.42.9 (28-Dec-2013)
/dev/sdb is entire device, not just one partition!
Proceed anyway? (y,n) y
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
2621440 inodes, 10485760 blocks
524288 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2157969408
320 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
```

图 7-2 格式化硬盘指定 inode 和 block

7.3 硬链接介绍

一般情况下，文件名和 inode 编号是一一对应的关系，每个 inode 号码对应一个文件名。但 UNIX/Linux 系统多个文件名也可以指向同一个 inode 号码。这意味着可以用不同的文件名访问同样的内容，对文件内容进行修改，会影响到所有文件名。但删除一个文件名，不影响另一个文件名的访问。这种情况被称为硬链接（hard link）。

创建硬链接的命令为 ln jf1.txt jf2.txt，其中 jf1.txt 为源文件，jf2.txt 为目标文件。如果上述命令源文件与目标文件的 inode 号码相同，则都指向同一个 inode。inode 信息中有一项叫作“链接数”，记录指向该 inode 的文件名总数，这时会增加 1，变成 2，如图 7-3 所示。

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# touch jf1.txt
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ll jf1.txt
-rw-r--r-- 1 root root 0 May  5 17:19 jf1.txt
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ln jf1.txt jf2.txt
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ll jf1.txt
-rw-r--r-- 2 root root 0 May  5 17:19 jf1.txt
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ll jf2.txt
-rw-r--r-- 2 root root 0 May  5 17:19 jf2.txt
[root@www-jfedu-net ~]#
```

图 7-3 jf1.txt jf2.txt 硬链接 inode 值变化

同理，删除一个 jf2.txt 文件，就会使得 jf1.txt inode 节点中的“链接数”减 1。如果该 inode 值减到 0，表明没有文件名指向这个 inode，系统就会回收这个 inode 号码，以及其所对应 block 区域，如图 7-4 所示。

实用小技巧：硬链接不能跨分区链接，硬链接只能对文件生效，对目录无效，也即是目录不能创建硬链接。硬链接源文件与目标文件共用一个 inode 值，从某种意义上来说，节省 inode 空间。不管是单独删除源文件还是删除目标文件，文件内容始终存在。链接后的文

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ll jf1.txt jf2.txt
-rw-r--r-- 2 root root 0 May  5 17:19 jf1.txt
-rw-r--r-- 2 root root 0 May  5 17:19 jf2.txt
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# rm -rf jf2.txt
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ll jf1.txt
-rw-r--r-- 1 root root 0 May  5 17:19 jf1.txt
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# 
```

图 7-4　删除 jf2.txt 硬链接 inode 值变化

件不占用系统多余的空间。

7.4　软链接介绍

除了硬链接以外，还有一种链接——软链接。文件 jf1.txt 和文件 jf2.txt 的 inode 号码虽然不同，但是文件 jf2.txt 的内容是文件 jf1.txt 的路径。读取文件 jf2.txt 时，系统会自动将访问者导向文件 jf1.txt。

无论打开哪一个文件，最终读取的文件都是 jf1.txt。这时，文件 jf2.txt 就称为文件 jf1.txt 的“软链接”(soft link)或者“符号链接”(symbolic link)。

文件 jf2.txt 依赖于文件 jf1.txt 而存在，如果删除了文件 jf1.txt，打开文件 jf2.txt 就会报错 No such file or directory。

软链接与硬链接最大的不同是文件 jf2.txt 指向文件 jf1.txt 的文件名，而不是文件 jf1.txt 的 inode 号码，因此文件 jf1.txt 的 inode 链接数不会发生变化，如图 7-5 所示。

```
[root@localhost ~]# ls -li jf1.txt
790403 -rw-r--r-- 1 root root 0 May  5 17:50 jf1.txt
[root@localhost ~]#
[root@localhost ~]# ln -s jf1.txt jf2.txt
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# ll -li jf1.txt jf2.txt
790403 -rw-r--r-- 1 root root 0 May  5 17:50 jf1.txt
795230 lrwxrwxrwx 1 root root 7 May  5 17:50 jf2.txt -> jf1.txt
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# rm -rf jf1.txt
[root@localhost ~]#
[root@localhost ~]# ll -li jf2.txt
795230 lrwxrwxrwx 1 root root 7 May  5 17:50 jf2.txt -> jf1.txt
[root@localhost ~]# 
```

图 7-5　删除 jf1.txt 源文件链接数不变

实用小技巧：软链接可以跨分区链接，软链接支持目录同时也支持文件的链接。软链接源文件与目标文件 inode 不相同，从某种意义上说，会消耗更多 inode 空间。不管是删除源文件还是重启系统，该软链接还存在，但是文件内容会丢失，一旦新建源同名文件名，软链接文件恢复正常。

7.5 Linux 下磁盘实战操作命令

企业真实场景由于硬盘常年大量读写，经常会出现坏盘，需要更换硬盘。或者由于磁盘空间不足，需添加新硬盘，新添加的硬盘需要经过格式化、分区才能被 Linux 系统所使用。虚拟机 CentOS 7 Linux 模拟 DELL R730 真实服务器添加一块新硬盘，不需要关机，直接插入用硬盘即可，一般硬盘均支持热插拔功能。企业中添加新硬盘的操作流程如下：

(1) 检测 Linux 系统识别的硬盘设备，新添加硬盘被识别为/dev/sdb，如果有多块硬盘，会依次识别成/dev/sdc、/dev/sdd 等设备名称，如图 7-6 所示。

```
fdisk -l
```

```
[root@www-jfedu-net ~]# fdisk -l

Disk /dev/sda: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x00077233

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048      411647      204800   83  Linux
/dev/sda2          411648     1460223      524288   82  Linux swap / Solar
/dev/sda3         1460224    78217215    38378496   83  Linux

Disk /dev/sdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

图 7-6 fdisk 查看 Linux 系统硬盘设备

(2) 基于新硬盘/dev/sdb 设备，创建磁盘分区/dev/sdb1，如图 7-7 所示。

```
fdisk /dev/sdb
```

```
[root@www-jfedu-net ~]# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0x3e5bdbfe.

Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   g   create a new empty GPT partition table
   G   create an IRIX (SGI) partition table
   l   list known partition types
   m   print this menu
```

图 7-7 fdisk /dev/sdb 分区

(3) fdisk 分区命令参数如下，常用参数包括 m、n、p、e、d、w。

- b：编辑 bsd disklabel。

- c：切换 dos 兼容性标志。
- d：删除一个分区。
- g：创建一个新的空 GPT 分区表。
- G：创建一个 IRIX(SGI)分区表。
- l：列出已知的分区类型。
- m：打印帮助菜单。
- n：添加一个新分区。
- o：创建一个新空 DOS 分区表。
- p：打印分区表信息。
- q：退出而不保存更改。
- s：创建一个新的空的 sun 磁盘标签。
- t：更改分区的系统 ID。
- u：更改显示/输入单位。
- v：验证分区表。
- w：将分区表写入磁盘并退出。
- x：额外功能。

(4) 创建/dev/sdb1 分区方法，执行命令 fdisk /dev/sdb，然后按屏幕提示依次输入 n、p、1，按 Enter 键，再输入+20G，按 Enter 键，输入 w，最后执行 fdisk -l| tail -10，如图 7-8 所示。

```
[root@www-jfedu-net ~]# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0x3cf5bf9c.

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-83886079, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-83886079, default 83886079): +20G
Partition 1 of type Linux and of size 20 GiB is set

Command (m for help): w
```

(a) fdisk /dev/sdb创建/dev/sdb1分区

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# fdisk -l|tail -10

Disk /dev/sdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x3cf5bf9c

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048    41945087    20971520   83  Linux
[root@www-jfedu-net ~]#
```

(b) fdisk -l查看/dev/sdb1分区

图 7-8 创建/dev/sdb1 分区

（5）mkfs. ext4　/dev/sdb1 格式化磁盘分区，如图 7-9 所示。

```
[root@www-jfedu-net ~]# mkfs.ext4 /dev/sdb1
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
1310720 inodes, 5242880 blocks
262144 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2153775104
160 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 265
        4096000
```

图 7-9　mkfs. ext4 格式化磁盘分区

（6）/dev/sdb1 分区格式化，使用 mount 命令挂载到/data/目录，命令详解如下，结果如图 7-10 所示。

- mkdir -p /data/：创建/data/数据目录。
- mount /dev/sdb1 /data：挂载/dev/sdb1 分区至/data/目录。
- df -h：查看磁盘分区详情。
- echo "mount /dev/sdb1 /data" >>/etc/rc. local：将挂载分区命令加入/etc/rc. local 开机启动。

```
[root@www-jfedu-net ~]# mkdir -p /data/
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# mount  /dev/sdb1   /data
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        37G  5.7G   31G  16% /
devtmpfs        484M     0  484M   0% /dev
tmpfs           493M     0  493M   0% /dev/shm
tmpfs           493M   13M  480M   3% /run
tmpfs           493M     0  493M   0% /sys/fs/cgroup
/dev/sda1       197M  103M   94M  53% /boot
tmpfs            99M     0   99M   0% /run/user/0
/dev/sdb1        20G   45M   19G   1% /data
[root@www-jfedu-net ~]# echo "mount  /dev/sdb1   /data" >>/etc/rc.local
```

图 7-10　mount 挂载/dev/sdb1 磁盘分区

（7）自动挂载分区除了可以加入到/etc/rc. local 开机启动之外，还可以加入到/etc/fstab 文件中，命令详解如下，结果如图 7-11 所示。

```
[root@www-jfedu-net ~]# vim /etc/fstab
#
# /etc/fstab
# Created by anaconda on Sat Aug 20 13:10:05 2016
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=01581e1f-c36c-4968-a5b8-7b570e424f1c /                       xfs     defaults
UUID=36ff9bf6-e8a4-48da-9bba-b9c7a89fe151 /boot                   xfs     defaults
UUID=fae320d7-1e4c-487d-87ea-8a79e1d0885a swap                    swap    defaults
/dev/sdb1                                 /data/                  ext4    defaults
```

图 7-11　/dev/sdb1 磁盘分区加入/etc/fstab 文件

```
/dev/sdb1 /data/ ext4 defaults 0 0
mount -o rw,remount /
```

如上命令表示重新挂载/系统，检测/etc/fstab 是否有误。

7.6 基于 GPT 格式磁盘分区

MBR 分区标准决定了 MBR 只支持在 2TB 以下的硬盘分区，为了能支持使用大于 2TB 硬盘空间，需使用 GPT 格式进行分区。创建大于 2TB 的分区，需使用 parted 工具。

在企业真实环境中，通常一台服务器有多块硬盘，整个硬盘容量为 10TB，需要基于 GTP 格式对 10TB 硬盘进行分区，操作步骤如下：

- parted -s /dev/sdb mklabel gpt：设置分区类型为 gpt 格式。
- mkfs. ext3 /dev/sdb：基于 ext3 文件系统类型格式化。
- mount /dev/sdb /data/：挂载/dev/sdb 设备至/data/目录。

(1) 如图 7-12 所示，假设/dev/sdb 为 10TB 硬盘，使用 GPT 格式来格式化磁盘。

```
/dev/sda3             536        6528    48126976

Disk /dev/sdb: 53.7 GB, 53687091200 bytes
255 heads, 63 sectors/track, 6527 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 b
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

[root@localhost ~]# fdisk -l
```

图 7-12 假设/dev/sdb 为 10TB 设备

(2) 执行命令 parted -s /dev/sdb mklabel gpt，如图 7-13 所示。

```
[root@localhost ~]# parted -s /dev/sdb  mklabel gpt
[root@localhost ~]#
[root@localhost ~]# fdisk -l |tail

WARNING: GPT (GUID Partition Table) detected on '/dev/sdb'! The
 Use GNU Parted.

Disk /dev/sdb: 53.7 GB, 53687091200 bytes
255 heads, 63 sectors/track, 6527 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        6528    52428799+  ee  GPT
[root@localhost ~]#
```

图 7-13 设置/dev/sdb 为 GPT 格式磁盘

(3) 基于 mkfs. ext3 /dev/sdb 格式化磁盘，如图 7-14 所示。

parted 命令行也可以进行分区，依次输入如下命令，如图 7-15 所示。

```
parted→select /dev/sdb→mklabel gpt→mkpart primary 0 - 1→print
mkfs.ext3 /dev/sdb1
mount /dev/sdb1 /data/
```

```
[root@localhost ~]# mkfs.ext3 /dev/sdb
mke2fs 1.41.12 (17-May-2010)
/dev/sdb is entire device, not just one partition!
无论如何也要继续? (y,n) y
文件系统标签=
操作系统:Linux
块大小=4096 (log=2)
分块大小=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
3276800 inodes, 13107200 blocks
655360 blocks (5.00%) reserved for the super user
第一个数据块=0
Maximum filesystem blocks=4294967296
```

图 7-14 格式/dev/sdb 磁盘

```
[root@localhost ~]# parted
GNU Parted 2.1
使用 /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of c
(parted) select /dev/sdb            选择/dev/sdb硬盘
使用 /dev/sdb
(parted)
(parted) mklabel gpt                格式类型为GPT
警告: The existing disk label on /dev/sdb will be dest
lost. Do you want to continue?
是/Yes/否/No? yes
(parted)
(parted) mkpart primary 0 -1        将整块硬盘分为一个分区
警告: The resulting partition is not properly aligned
忽略/Ignore/放弃/Cancel?
忽略/Ignore/放弃/Cancel? ignore
(parted)
(parted) print                      打印我们刚分区的磁盘信息
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sdb: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
```

(a) parted工具执行GPT格式分区(1)

```
[root@localhost ~]# mkfs.ext3  /dev/sdb1
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
1310720 inodes, 5242631 blocks
262131 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
160 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632
```

(b) parted工具执行GPT格式分区(2)

```
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# mount      /dev/sdb1     /data/
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        30G  4.8G   23G  18% /
tmpfs           242M     0  242M   0% /dev/shm
/dev/sda1       194M   44M  141M  24% /boot
/dev/sdb1        20G  173M   19G   1% /data
[root@localhost ~]#
[root@localhost ~]#
```

(c) parted工具执行GPT格式分区(3)

图 7-15 parted 工具执行 GPT 格式分区

7.7 mount 命令工具

mount 命令工具主要用于将设备或者分区挂载至 Linux 系统目录下，Linux 系统在分区时，基于 mount 机制将/dev/sda 分区挂载至系统目录，将设备与目录挂载之后，Linux 操作系统方可进行文件的读取和存储。

7.7.1 mount 命令参数详解

以下为企业中 mount 命令常用参数详解。

常见用法如下：

```
mount [-Vh]
mount -a [-fFnrsvw] [-t vfstype]
mount [-fnrsvw] [-o options [,...]] device | dir
mount [-fnrsvw] [-t vfstype] [-o options] device dir
```

参数详解如下：

- -V：显示 mount 工具版本号。
- -l：显示已加载的文件系统列表。
- -h：显示帮助信息并退出。
- -v：输出指令执行的详细信息。
- -n：加载没有写入文件/etc/mtab 中的文件系统。
- -r：将文件系统加载为只读模式。
- -a：加载文件/etc/fstab 中配置的所有文件系统。
- -o：指定 mount 挂载扩展参数，常见扩展指令 rw、remount、loop 等。

其中与-o 相关指令如下：

- -o atime：系统会在每次读取文档时更新文档时间。
- -o noatime：系统会在每次读取文档时不更新文档时间。
- -o defaults：使用预设的选项 rw，suid，dev，exec，auto，nouser 等。
- -o exec：允许执行档被执行。
- -o user、-o nouser：使用者可以执行 mount/umount 的动作。
- -o remount：将已挂载的系统分区重新以其他再次模式挂载。
- -o ro：只读模式挂载。
- -o rw：可读可写模式挂载。
- -o loop：使用 loop 模式，把文件当成设备挂载至系统目录。
- -t：指定 mount 挂载设备类型，常见类型有 nfs、ntfs-3g、vfat、iso9660 等。

其中与-t 相关指令如下：

- iso9660：光盘或光盘镜像。

- msdos：Fat16 文件系统。
- vfat：Fat32 文件系统。
- ntfs：ntfs 文件系统。
- ntfs-3g：识别移动硬盘格式。
- smbfs：挂载 Windows 文件网络共享。
- nfs：UNIX/Linux 文件网络共享。

7.7.2 企业常用 mount 案例

mount 常用案例演示详解如下：

- mount/dev/sdb1/data：挂载/dev/sdb1 分区至/data/目录。
- mount/dev/cdrom/mnt：挂载 cdrom 光盘至/mnt 目录。
- mount -t ntfs-3g/dev/sdc/data1：挂载/dev/sdc 移动硬盘至/data1 目录。
- mount -o remount,rw/：重新以读写模式挂载/系统。
- mount -t iso9660 -o loop centos7.iso /mnt：将 CentOS 7.iso 镜像文件挂载至/mnt 目录。
- mount -t fat32/dev/sdd1/mnt：将 U 盘/dev/sdd1 挂载至/mnt/目录。
- mount -t nfs 192.168.1.11:/data//mnt：将远程 192.168.1.11:/data 目录挂载至本地/mnt 目录。

7.8 Linux 硬盘故障修复

企业服务器运维中，经常会发现操作系统的分区变成只读文件系统，错误提示信息为 Read-only file system。出现只读文件系统会导致只能读取而无法写入新文件、新数据等操作。

造成该问题的原因包括：磁盘老旧长期大量的读写、文件系统文件被破坏、磁盘碎片文件、异常断电、读写中断等。

以企业 CentOS 7 Linux 为案例来修复文件系统，步骤如下：

(1) 远程备份本地其他重要数据，出现只读文件系统，需先备份其他重要数据，基于 rsync|scp 远程备份，其中/data 为源目录，/data/backup/2017/为目标备份目录。

```
rsync -av /data/ root@192.168.111.188:/data/backup/2017/
```

(2) 可以重新挂载/系统，挂载命令如下，测试文件系统是否可以写入文件。

```
mount -o remount,rw /
```

(3) 如果重新挂载/系统无法解决问题，则需重启服务器以 CD/DVD 光盘引导进入 Linux Rescue 修复模式。如图 7-16 所示，光标选择 Troubleshooting，按 Enter 键，然后选择 Rescue a CentOS system，按 Enter 键。



(a) 光盘引导进入修复模式(1)

(b) 光盘引导进入修复模式(2)

图 7-16　光盘引导进入修复模式

（4）选择“1）Continue”继续操作，如图 7-17 所示。

```
The rescue environment will now attempt to find your Linux installation and moun
t it under the directory : /mnt/sysimage.  You can then make any changes require
d to your system.  Choose '1' to proceed with this step.
You can choose to mount your file systems read-only instead of read-write by cho
osing '2'.
If for some reason this process does not work choose '3' to skip directly to a s
hell.

 1) Continue

 2) Read-only mount

 3) Skip to shell

 4) Quit (Reboot)

Please make a selection from the above: _
```

图 7-17　选择 Continue 继续进入系统

（5）进入修复模式，执行如下命令，df -h 显示原来的文件系统，如图 7-18 所示。

```
chroot /mnt/sysimage
df  - h
```

```
sh-4.2# chroot  /mnt/sysimage/


bash-4.2#
bash-4.2#
bash-4.2#
bash-4.2# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        37G  5.7G   31G  16% /
devtmpfs        474M     0  474M   0% /dev
tmpfs           493M     0  493M   0% /dev/shm
tmpfs           493M   13M  481M   3% /run
/dev/sda1       197M  103M   94M  53% /boot
/dev/sdb1        20G   45M   19G   1% /data
bash-4.2#
```

图 7-18　切换原分区目录

（6）对有异常的分区进行检测并修复，根据文件系统类型，执行命令如下：

```
umount /dev/sda3
fsck.ext4 /dev/sda3 -y
```

（7）修复完成之后，重启系统即可。

```
reboot
```

本章小结

通过对本章内容的学习，读者掌握了 Linux 硬盘内部结构、block 及 inode 特性，能够对企业硬盘进行分区、格式化等操作，满足企业的日常需求。

基于 mount 工具，能对硬盘、各类文件系统进行挂载操作，同时对只读文件系统能快速修复并投入使用。

同步作业

1. 软链接与硬链接的区别是什么？

2. 有一块 4TB 的移动硬盘，如何将数据复制至服务器/data/目录，服务器空间为 10TB，请写出详细步骤。

3. 运维部小刘发现公司 IDC 机房一台 DELL R730 服务器，/data/目录不可写，而/boot 目录可读可写，请问原因是什么，如何修复/data/目录？

4. 公司一台 DELL R730 服务器/data/images 目录存放了大量的小文件，运维人员向该目录写入 1MB 测试文件，提示磁盘空间不足，而通过 df -h 显示剩余可用空间为 500GB，请问是什么原因导致的，如何解决该问题？

5. 机房一台 DELL R730 服务器，由于业务需求临时重启，重启完 20 分钟后还无法登录系统，检查控制台输出，一直卡在 MySQL 服务启动项，请问如何快速解决让系统正常启动，请写出解决步骤。

第 8 章



Linux 文件服务器企业实战

运维和管理企业 Linux 服务器，除了要熟练掌握 Linux 系统本身的维护和管理之外，最重要的是熟练甚至精通 Linux 系统安装配置各种应用软件，对软件进行调优以及针对软件在使用中遇到的各类问题，能够快速定位并解决。

本章向读者介绍进程、线程、企业 Vsftpd 服务器实战、匿名用户访问、系统用户访问及虚拟用户实战等内容。

8.1　进程与线程的概念及区别

Linux 系统各种软件和服务存在于系统，必然会占用系统各种资源，系统资源是如何分配及调度的，本节将给读者展示系统进程、资源及调度相关的内容。

进程(process)是计算机中的软件程序关于某数据集合上的一次运行活动，是系统进行资源分配和调度的基本单位，是操作系统结构的基础。

在早期面向进程设计的计算机结构中，进程是程序的基本执行实体，在当代面向线程设计的计算机结构中，进程是线程的容器。软件程序是对指令、数据及其组织形式的描述，而进程是程序的实体，通常而言，把运行在系统中的软件程序称之为进程。

除了进程，读者通常会听到线程的概念，线程也被称为轻量级进程(lightweight process，LWP)，是程序执行流的最小单元。一个标准的线程由线程 ID，当前指令指针(PC)，寄存器集合和堆栈组成。

线程是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有操作系统资源，但是该线程可与同属进程的其他线程共享该进程所拥有的全部资源。

程序、进程、线程三者区别如下。

- 程序：程序并不能单独执行，是静止的，只有将程序加载到内存中，系统为其分配资源后才能够执行。
- 进程：程序对一个数据集的动态执行过程，一个进程包含一个或者更多的线程，一个线程同时只能被一个进程所拥有，进程是分配资源的基本单位。进程拥有独立的内

存单元,而多个线程共享内存,从而提高了应用程序的运行效率。

- 线程:线程是进程内的基本调度单位,线程的划分尺度小于进程,并发性更高,线程本身不拥有系统资源,但是该线程可与同属进程的其他线程共享该进程所拥有的全部资源。每一个独立的线程,都有一个程序运行的入口、顺序执行序列和程序的出口。

程序、进程、线程三者的关系拓扑图如图 8-1 所示。

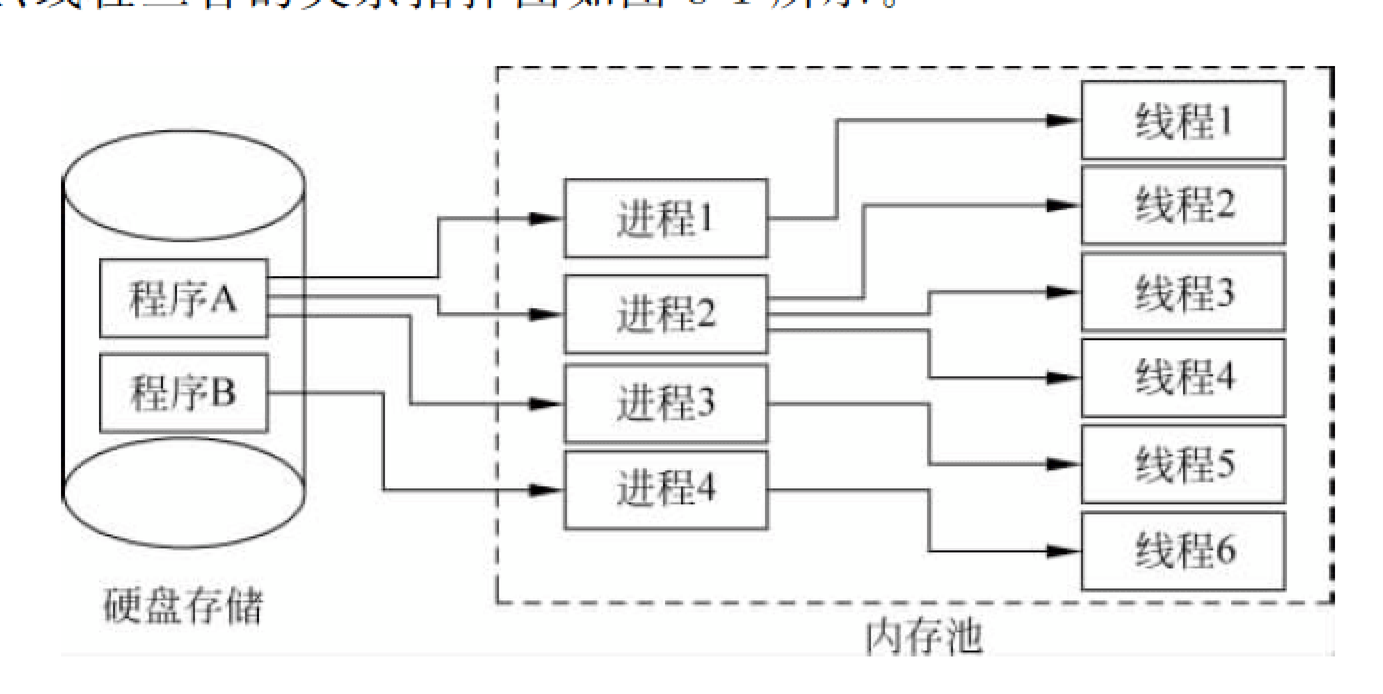


图 8-1　程序、进程、线程三者的关系拓扑图

如图 8-1 所示,多进程、多线程的区别如下。

- 多进程:每个进程互相独立,不影响主程序的稳定性,某个子进程崩溃对其他进程没有影响,通过增加 CPU 可以扩充软件的性能,可以减少线程加锁/解锁的影响,极大提高性能。缺点是多进程逻辑控制复杂,需要和主程序交互,需要跨进程边界,进程之间上下文切换比线程之间上下文切换代价大。
- 多线程:无须跨进程,程序逻辑和控制方式简单,所有线程共享该进程的内存和变量等。缺点是每个线程与主程序共用地址空间,线程之间的同步和加锁控制比较麻烦,一个线程的崩溃会影响到整个进程或者程序的稳定性。

8.2　Vsftpd 服务器企业实战

文件传输协议(file transfer protocol,FTP),基于该协议 FTP 客户端与服务端可以实现共享文件、上传文件、下载文件。FTP 基于 TCP 协议生成一个虚拟的连接,主要用于控制 FTP 连接信息,同时再生成一个单独的 TCP 连接用于 FTP 数据传输。用户可以通过客户端向 FTP 服务器端上传、下载、删除文件,FTP 服务器端可以同时提供给多人共享使用。

FTP 服务是 client/server(简称 C/S)模式,基于 FTP 协议实现 FTP 文件对外共享及传输的软件称之为 FTP 服务器源端,客户端程序基于 FTP 协议,则称之为 FTP 客户端,FTP 客户端可以向 FTP 服务器上传、下载文件。

8.2.1　FTP 传输模式

FTP 基于 C/S 模式,FTP 客户端与服务器端有两种传输模式,分别是 FTP 主动模式、

FTP 被动模式，主被动模式均是以 FTP 服务器端为参照。FTP 传输模式如图 8-2 所示，其中图 8-2(a)为主动模式，图 8-2(b)为被动模式。主被动模式详细区别如下。

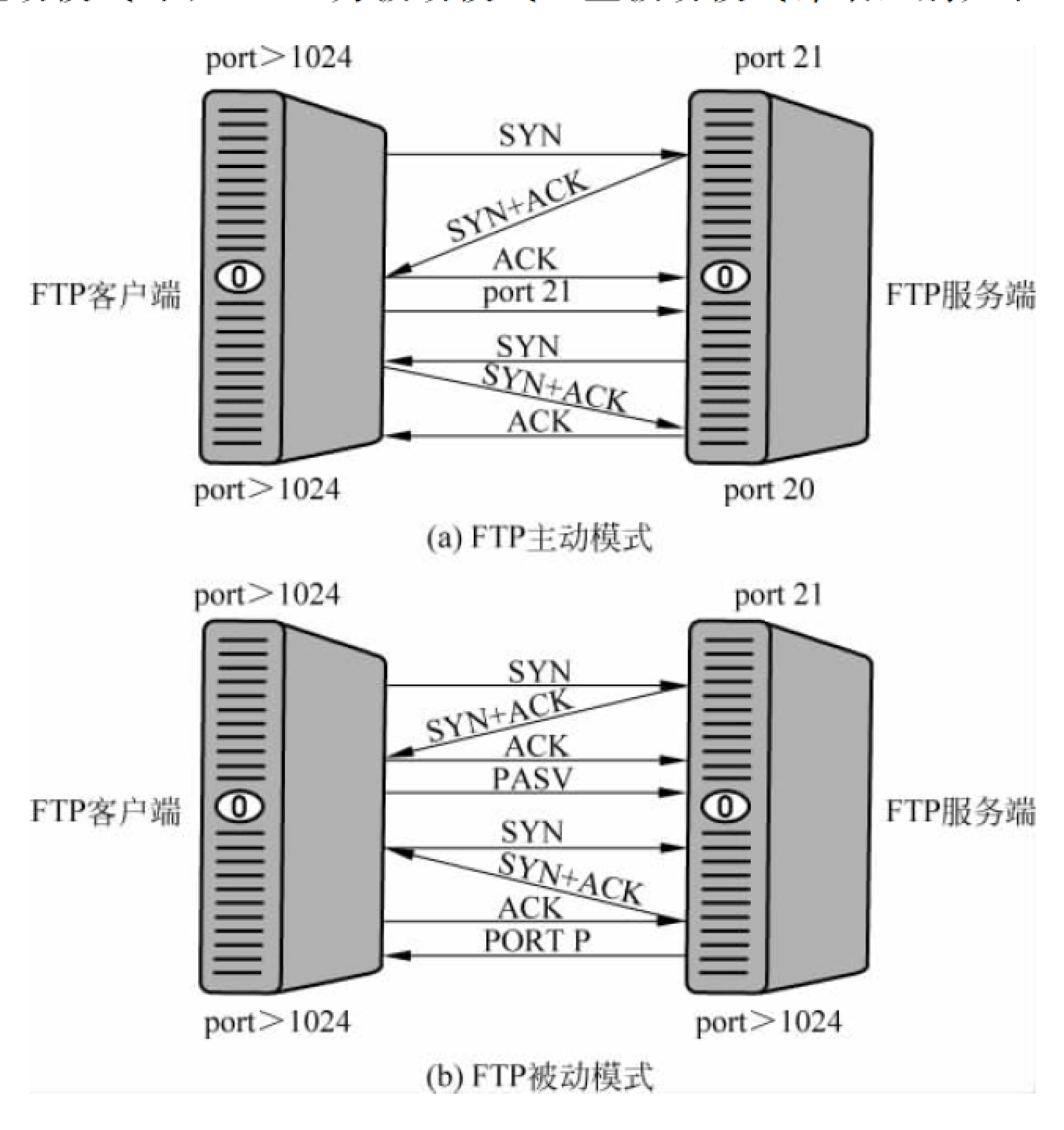


图 8-2　FTP 传输模式

- FTP 主动模式：客户端从一个任意的端口 N(N>1024)连接到 FTP 服务器的 port 21 命令端口，客户端开始监听端口 N+1，并发送 FTP 命令"port N+1"到 FTP 服务器，FTP 服务器以数据端口(20)连接到客户端指定的数据端口(N+1)。
- FTP 被动模式：客户端从一个任意的端口 N(N>1024)连接到 FTP 服务器的 port 21 命令端口，客户端开始监听端口 N+1，客户端提交 PASV 命令，服务器会开启一个任意的端口(P>1024)，并发送 PORT P 命令给客户端。客户端发起从本地端口 N+1 到服务器的端口 P 的连接用来传送数据。

在企业实际环境中，如果 FTP 客户端与 FTP 服务端均开放防火墙，FTP 需以主动模式工作，这样只需要在 FTP 服务器端防火墙规则中，开放 20、21 端口即可。

8.2.2　Vsftpd 服务器简介

目前主流的 FTP 服务器端软件包括 Vsftpd、ProFTPD、PureFTPd、Wuftpd、Server-U FTP、FileZilla Server 等，其中 UNIX/Linux 使用较为广泛的 FTP 服务器端软件为 Vsftpd。

非常安全的 FTP 服务进程(very secure FTP daemon，Vsftpd)，Vsftpd 是在 UNIX/

Linux 发行版中最主流的 FTP 服务器程序，优点是小巧轻快、安全易用、稳定高效、满足企业跨部门、多用户的使用等。

Vsftpd 基于 GPL 开源协议发布，在中小企业中得到广泛的应用。Vsftpd 可以快速上手，基于 Vsftpd 虚拟用户方式，访问验证更加安全。Vsftpd 还可以基于 MySQL 数据库作安全验证，多重安全防护。

8.2.3 Vsftpd 服务器安装配置

Vsftpd 服务器端安装有两种方法：一是基于 YUM 方式安装；二是基于源码编译安装，最终实现效果完全一样，本文采用 YUM 安装 Vsftpd，具体步骤如下：

（1）在 shell 命令行执行如下命令，如图 8-3 所示。

```
yum install vsftpd* -y
```

```
[root@www-jfedu-net ~]# yum install vsftpd* -y
已加载插件：fastestmirror, product-id, search-disabled-repos, subscrip
This system is not registered with Subscription Management. You can us
base
extras
updates
updates/7/x86_64/primary_db
Loading mirror speeds from cached hostfile
 * base: mirrors.btte.net
 * extras: centos.ustc.edu.cn
 * updates: ftp.jaist.ac.jp
正在解决依赖关系
--> 正在检查事务
---> 软件包 vsftpd.x86_64.0.3.0.2-11.el7_2 将被 升级
---> 软件包 vsftpd.x86_64.0.3.0.2-21.el7 将被 更新
---> 软件包 vsftpd-sysvinit.x86_64.0.3.0.2-21.el7 将被 安装
```

图 8-3 YUM 安装 Vsftpd 服务端

（2）Vsftpd 安装后的配置文件路径、启动 Vsftpd 服务及查看进程是否启动，如图 8-4 所示。

```
rpm -ql vsftpd|more
systemctl restart vsftpd.service
ps -ef |grep vsftpd
```

```
[root@www-jfedu-net ~]# rpm -ql vsftpd|more
/etc/logrotate.d/vsftpd
/etc/pam.d/vsftpd
/etc/vsftpd
/etc/vsftpd/ftpusers
/etc/vsftpd/user_list
/etc/vsftpd/vsftpd.conf
/etc/vsftpd/vsftpd_conf_migrate.sh
/usr/lib/systemd/system-generators/vsftpd-generator
/usr/lib/systemd/system/vsftpd.service
/usr/lib/systemd/system/vsftpd.target
/usr/lib/systemd/system/vsftpd@.service
/usr/sbin/vsftpd
/usr/share/doc/vsftpd-3.0.2
/usr/share/doc/vsftpd-3.0.2/AUDIT
/usr/share/doc/vsftpd-3.0.2/BENCHMARKS
```

图 8-4 打印 Vsftpd 软件安装后路径

（3）Vsftpd.conf 默认配置文件详解如下：

- anonymous_enable=YES：开启匿名用户访问。
- local_enable=YES：启用本地系统用户访问。
- write_enable=YES：本地系统用户写入权限。
- local_umask=022：本地用户创建文件及目录默认权限掩码。
- dirmessage_enable=YES：打印目录显示信息，通常用于用户第一次访问目录时，信息提示。
- xferlog_enable=YES：启用上传/下载日志记录。
- connect_from_port_20=YES FTP：使用 20 端口进行数据传输。
- xferlog_std_format=YES：日志文件将根据 xferlog 的标准格式写入。
- listen=NO：Vsftpd 不以独立的服务启动，通过 Xinetd 服务管理，建议改成 YES。
- listen_ipv6=YES：启用 IPv6 监听。
- pam_service_name=vsftpd：登录 FTP 服务器，依据/etc/pam.d/vsftpd 中内容进行认证。
- userlist_enable=YES：vsftpd.user_list 和 ftpusers 配置文件里用户禁止访问 FTP。
- tcp_wrappers=YES：设置 Vsftpd 与 tcp wrapper 结合进行主机的访问控制，Vsftpd 服务器检查/etc/hosts.allow 和/etc/hosts.deny 中的设置来决定请求连接的主机，是否允许访问该 FTP 服务器。

（4）启动 Vsftpd 服务后，通过 Windows 客户端资源管理器访问 Vsftpd 服务器端，如图 8-5 所示。

```
ftp://192.168.111.131/
```

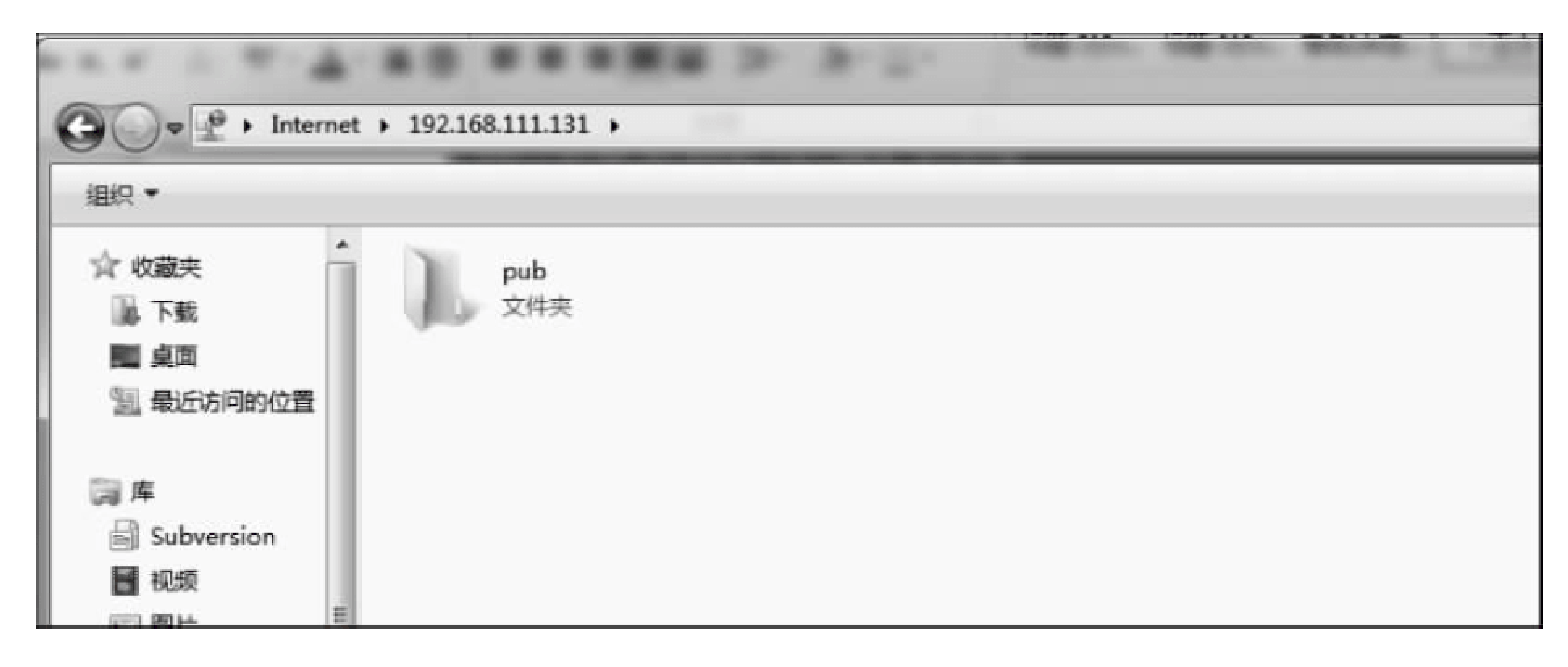


图 8-5　匿名用户访问 FTP 默认目录

FTP 主被动模式的选择，默认为主动模式，设置为被动模式的方法如下：

```
pasv_enable = YES
```

```
pasv_min_port = 60000
pasv_max_port = 60100
```

8.2.4 Vsftpd 匿名用户配置

Vsftpd 默认以匿名用户访问，匿名用户默认访问的 FTP 服务器发布端路径为/var/ftp/pub，匿名用户只有查看权限，无法创建、删除、修改。如需关闭 FTP 匿名用户访问，需修改配置文件/etc/vsftpd/vsftpd.conf，将 anonymous_enable=YES 修改为 anonymous_enable=NO，重启 Vsftpd 服务即可。

如果允许匿名用户能够上传、下载、删除文件，需在/etc/vsftpd/vsftpd.conf 配置文件中加入以下代码，详解如下：

- anon_upload_enable=YES：允许匿名用户上传文件。
- anon_mkdir_write_enable=YES：允许匿名用户创建目录。
- anon_other_write_enable=YES：允许匿名用户其他写入权限。

匿名用户完整的 vsftpd.conf 配置文件代码如下：

```
anonymous_enable = YES
local_enable = YES
write_enable = YES
local_umask = 022
anon_upload_enable = YES
anon_mkdir_write_enable = YES
anon_other_write_enable = YES
dirmessage_enable = YES
xferlog_enable = YES
connect_from_port_20 = YES
xferlog_std_format = YES
listen = NO
listen_ipv6 = YES
pam_service_name = vsftpd
userlist_enable = YES
tcp_wrappers = YES
```

由于默认 Vsftpd 匿名用户有两种：anonymous、ftp，所以匿名用户如果需要上传文件、删除及修改等权限，需要 Vsftpd 用户对/var/ftp/pub 目录有写入权限，使用 chown 和 chmod 任意一种命令均可设置权限，具体设置命令如下：

```
chown  - R ftp pub/
chmod o + w pub/
```

如上 vsftpd.conf 配置文件配置完毕，同时权限设置完毕，重启 Vsftpd 服务即可，通过 Windows 客户端访问，能够上传文件、删除文件、创建目录等操作，如图 8-6 所示。



图 8-6 匿名用户访问上传文件

8.2.5 Vsftpd 系统用户配置

Vsftpd 匿名用户设置完毕，任何人都可以查看 FTP 服务器端的文件、目录，甚至可以修改、删除、文件和目录，如何存放私密文件在 FTP 服务器端，并保证文件或者目录专属于拥有者呢？Vsftpd 系统用户可以实现该需求，解决上述问题。

实现 Vsftpd 系统用户方式验证，只需在 Linux 系统中创建多个用户即可，创建用户使用 useradd 指令，同时给用户设置密码，即可通过用户和密码登录 FTP，进行文件上传、下载、删除等操作。Vsftpd 系统用户实现方法步骤如下：

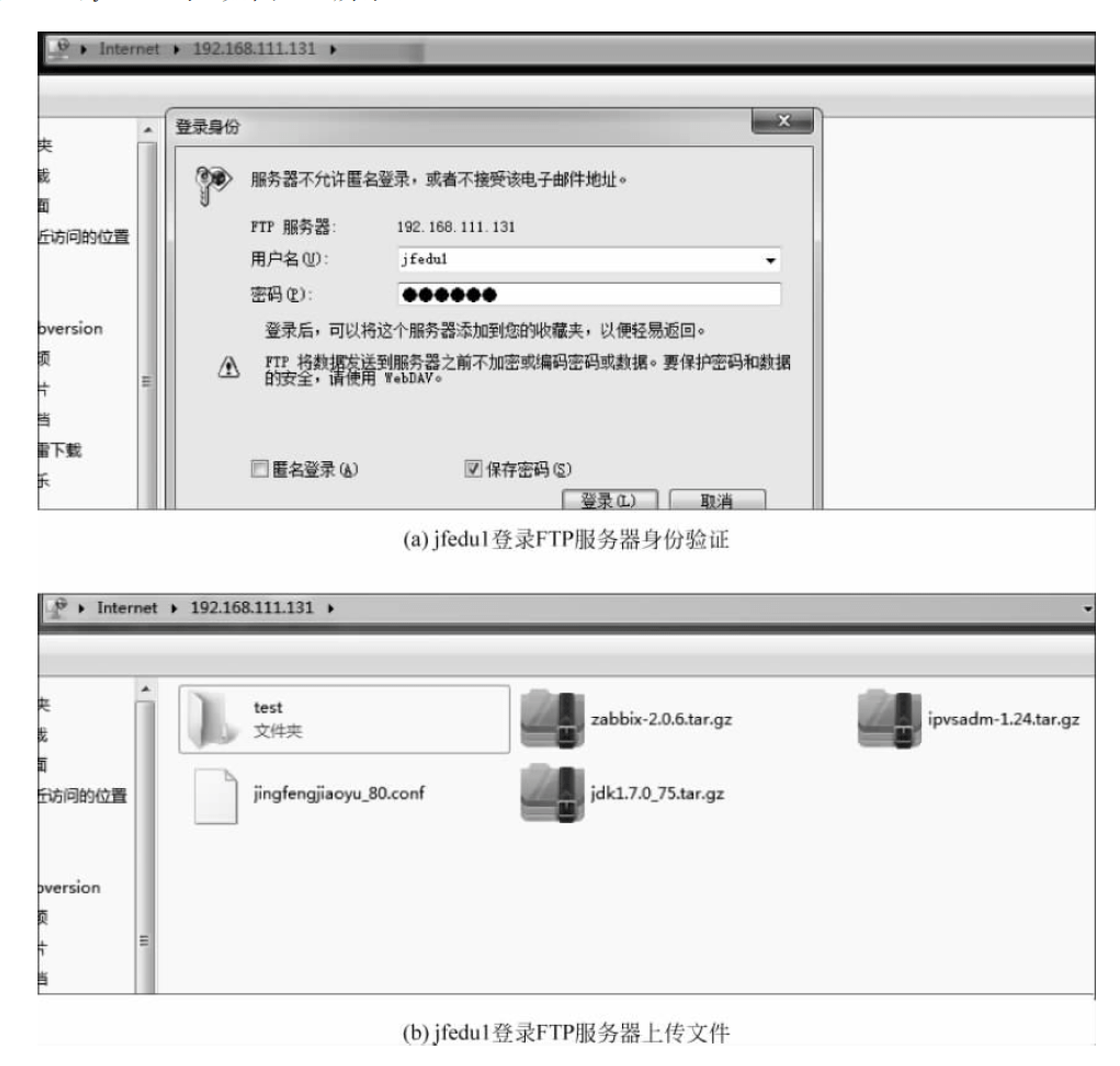
（1）Linux 系统中创建系统用户 jfedu1、jfedu2，分别设置密码为 123456。

```
useradd jfedu1
useradd jfedu2
echo 123456|passwd --stdin jfedu1
echo 123456|passwd --stdin jfedu2
```

（2）修改 vsftpd.conf 配置文件代码如下：

```
anonymous_enable = NO
local_enable = YES
write_enable = YES
local_umask = 022
dirmessage_enable = YES
xferlog_enable = YES
connect_from_port_20 = YES
xferlog_std_format = YES
listen = NO
listen_ipv6 = YES
pam_service_name = vsftpd
userlist_enable = YES
tcp_wrappers = YES
```

(3) 通过 Windows 资源客户端验证,使用 jfedu1、jfedu2 用户登录 FTP 服务器,即可上传文件、删除文件、下载文件,jfedu1、jfedu2 系统用户上传文件的家目录在/home/jfedu1、/home/jfedu2 下,如图 8-7 所示。



(a) jfedu1登录FTP服务器身份验证

(b) jfedu1登录FTP服务器上传文件

图 8-7　jfedu1 用户登录 FTP 服务器

8.2.6　Vsftpd 虚拟用户配置

Vsftpd 基于系统用户访问 FTP 服务器,系统用户越多越不利于管理,而且不利于系统安全,为了能更加安全使用 Vsftpd,可以使用 Vsftpd 虚拟用户方式。

Vsftpd 虚拟用户原理为虚拟用户没有实际的真实系统用户,而是通过映射到其中一个真实用户以及设置相应的权限来实现访问验证,虚拟用户不能登录 Linux 系统,从而让系统更加的安全可靠。

Vsftpd 虚拟用户企业案例配置步骤如下：

(1) 安装 Vsftpd 虚拟用户需要用到的软件及认证模块。

```
yum install pam* libdb-utils libdb* --skip-broken -y
```

(2) 创建虚拟用户临时文件/etc/vsftpd/ftpusers.txt，新建虚拟用户和密码，其中 jfedu001、jfedu002 为虚拟用户名，123456 为密码，如果有多个用户，依此格式填写即可。

```
jfedu001
123456
jfedu002
123456
```

(3) 生成 Vsftpd 虚拟用户数据库认证文件，设置权限为 700。

```
db_load -T -t hash -f /etc/vsftpd/ftpusers.txt /etc/vsftpd/vsftpd_login.db
chmod 700 /etc/vsftpd/vsftpd_login.db
```

(4) 配置 PAM 认证文件，/etc/pam.d/vsftpd 行首加入如下两行代码：

```
auth required pam_userdb.so db=/etc/vsftpd/vsftpd_login
account required pam_userdb.so db=/etc/vsftpd/vsftpd_login
```

(5) Vsftpd 虚拟用户需要映射到一个系统用户，该系统用户不需要密码，也不需要登录，主要用于虚拟用户映射使用，创建用户命令如下：

```
useradd -s /sbin/nologin ftpuser
```

(6) 完整的 vsftpd.conf 配置文件代码如下：

```
#global config Vsftpd 2017
anonymous_enable=YES
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
listen=NO
listen_ipv6=YES
userlist_enable=YES
tcp_wrappers=YES
#config virtual user FTP
pam_service_name=vsftpd
guest_enable=YES
guest_username=ftpuser
user_config_dir=/etc/vsftpd/vsftpd_user_conf
virtual_use_local_privs=YES
```

Vsftpd 虚拟用户配置文件参数详解如下：

- pam_service_name=vsftpd：虚拟用户启用 pam 认证。
- guest_enable=YES：启用虚拟用户。
- guest_username=ftpuser：映射虚拟用户至系统用户 ftpuser。
- user_config_dir=/etc/vsftpd/vsftpd_user_conf：设置虚拟用户配置文件所在的目录。
- virtual_use_local_privs=YES：虚拟用户使用与本地用户相同的权限。

（7）至此，所有虚拟用户共同使用/home/ftpuser 主目录实现文件上传与下载，可以在/etc/vsftpd/vsftpd_user_conf 目录创建虚拟用户各自的配置文件，创建虚拟用户配置文件主目录，代码如下：

```
mkdir -p /etc/vsftpd/vsftpd_user_conf/
```

（8）以下分别为虚拟用户 jfedu001、jfedu002 创建配置文件。

vim /etc/vsftpd/vsftpd_user_conf/jfedu001，同时创建私有的虚拟目录，代码如下：

```
local_root = /home/ftpuser/jfedu001
write_enable = YES
anon_world_readable_only = YES
anon_upload_enable = YES
anon_mkdir_write_enable = YES
anon_other_write_enable = YES
```

vim /etc/vsftpd/vsftpd_user_conf/jfedu002，同时创建私有的虚拟目录，代码如下：

```
local_root = /home/ftpuser/jfedu002
write_enable = YES
anon_world_readable_only = YES
anon_upload_enable = YES
anon_mkdir_write_enable = YES
anon_other_write_enable = YES
```

虚拟用户配置文件内容详解如下：

- local_root=/home/ftpuser/jfedu002：jfedu002 虚拟用户配置文件路径。
- write_enable=YES：允许登录用户有写权限。
- anon_world_readable_only=YES：允许匿名用户下载，然后读取文件。
- anon_upload_enable=YES：允许匿名用户上传文件权限，只有在 write_enable=YES 时该参数才生效。
- anon_mkdir_write_enable=YES：允许匿名用户创建目录，只有在 write_enable=YES 时该参数才生效。
- anon_other_write_enable=YES：允许匿名用户其他权限，例如删除、重命名等。

（9）创建虚拟用户各自虚拟目录，代码如下：

```
mkdir -p /home/ftpuser/{jfedu001,jfedu002} ; chown -R ftpuser:ftpuser /home/ftpuser
```

重启 Vsftpd 服务，通过 Windows 客户端资源管理器登录 Vsftpd 服务端，测试结果如图 8-8 所示。



(a) jfedu001虚拟用户登录FTP服务器



(b) jfedu001虚拟用户上传下载文件

图 8-8　测试结果

第二篇　Linux 进阶篇

Linux 进阶篇总共包含 6 个章节，第 9～14 章学习内容分别为 HTTP 协议详解、Apache Web 服务器企业实战、MySQL 服务器企业实战、LAMP 企业架构实战、Zabbix 分布式监控企业实战、Nginx Web 服务器企业实战。

读者通过对进阶篇 6 个章节的深入学习，可以在基础篇学习的 Linux 操作系统管理的基础上，快速上手、独立维护和管理企业各种服务。例如主流的 Apache、Nginx Web 服务器，并可以通过深入学习 HTTP 协议，掌握 HTTP 底层通信原理等。

同时读者能熟练构建企业级数据库管理集群，MySQL 主从复制、一主多从、读写分离实战保证网站数据的完整，对数据库配置文件进行调优、增加索引提供数据查询效率，如果数据库异常或缓慢，可以基于 MySQL 慢查询日志定位慢 SQL。

进阶篇引入 Redis 高性能缓存服务器，各大互联网公司都在使用 Redis。熟练掌握 Redis 对升职加薪及提升网站性能有巨大帮助，Redis 缓存还可以提高用户访问 Web 网站的效率，增强用户体验。同时随着企业服务器不断增加，基于 Zabbix 分布式的监控系统能够实时监控服务器 CPU、内存、硬盘、网卡及服务器上各种应用，做到有故障第一时间给相关人员发送微信报警，并第一时间处理问题。

互联网主流 Web 服务器软件 Nginx，得到各大企业 SA 的青睐，应用非常广泛，因此深入掌握 Nginx，对运维能力的提升帮助是非常大的。通过对进阶篇中 Nginx 的深入学习，读者能够熟练掌握 Nginx 的工作原理、安装配置、管理升级、负载均衡、动静分离、虚拟主机、参数调优、Nginx location、Nginx rewrite、日志切割、防盗链、HTTPS 等核心技术，更好地维护生产环境 Nginx 高性能 Web 服务器。

第 9 章



HTTP 协议详解

超文本传输协议(hypertext transfer protocol,HTTP)是在互联网上应用最为广泛的一种网络协议。所有的 WWW 服务器都基于该协议。HTTP 设计最初的目的是提供一种发布 Web 页面和接收 Web 页面的方法。

本章向读者介绍 TCP、HTTP 协议、HTTP 资源定位、HTTP 请求及响应头详细信息、HTTP 状态码及 MIME 类型详解等内容。

9.1 TCP 协议与 HTTP 协议

1960 年,美国人 Ted Nelson 构思了一种通过计算机处理文本信息的方法,并称之为超文本(hypertext),为 HTTP 超文本传输协议标准架构的发展奠定了根基。Ted Nelson 组织协调万维网协会(World Wide Web Consortium)和互联网工程工作小组(Internet Engineering Task Force)共同合作研究,最终发布了一系列的 RFC,其中著名的 RFC 2616 定义了 HTTP 1.1。

很多读者对 TCP 协议与 HTTP 协议存在疑问,这两者有何区别,从应用领域来说,TCP 协议主要用于数据传输控制,而 HTTP 协议主要用于应用层面的数据交互,本质上两者没有可比性。

HTTP 协议属于应用层协议,是建立在 TCP 协议基础之上的,HTTP 协议以客户端请求和服务器端应答为标准,浏览器通常称为客户端,而 Web 服务器称之为服务器端。客户端打开任意一个端口向服务端的指定端口(默认为 80)发起 HTTP 请求,首先会发起 TCP 三次握手,TCP 三次握手的目的是建立可靠的数据连接通道,TCP 三次握手通道建立完毕,进行 HTTP 数据交互,如图 9-1 所示。

当客户端请求的数据接收完毕后,HTTP 服务器端会断开 TCP 连接,整个 HTTP 连接过程非常短。HTTP 连接也称为无状态的连接,无状态连接是指客户端每次向服务器发起 HTTP 请求时,每次请求都会建立一个新的 HTTP 连接,而不是在一个 HTTP 请求基础上进行所有数据的交互。

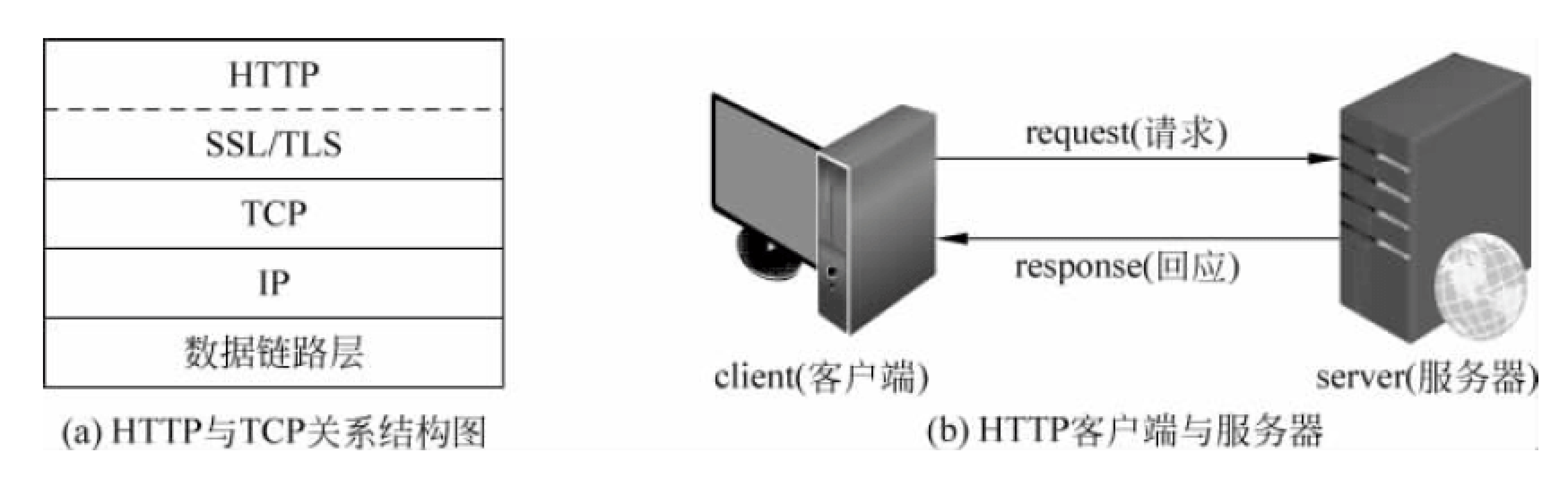


图 9-1　HTTP、TCP 协议关系

9.2　资源定位标识符

HTTP 请求的内容资源由统一资源标示符(uniform resource identifiers,URI)来标识，关于资源定位及标识有三种：URI、URN、URL，三种资源定位详解如下：

- 统一资源标识符(uniform resource identifier,URI)，用来唯一标识一个资源；
- 统一资源定位器(uniform resource locator,URL)，是一种具体的 URI,URL 可以用来标识一个资源，而且可以访问或者获取该资源；
- 统一资源命名(uniform resource name,URN)，通过名字来标识或识别资源。

如图 9-2 所示，可以直观区分 URI、URN、URL 的区别。

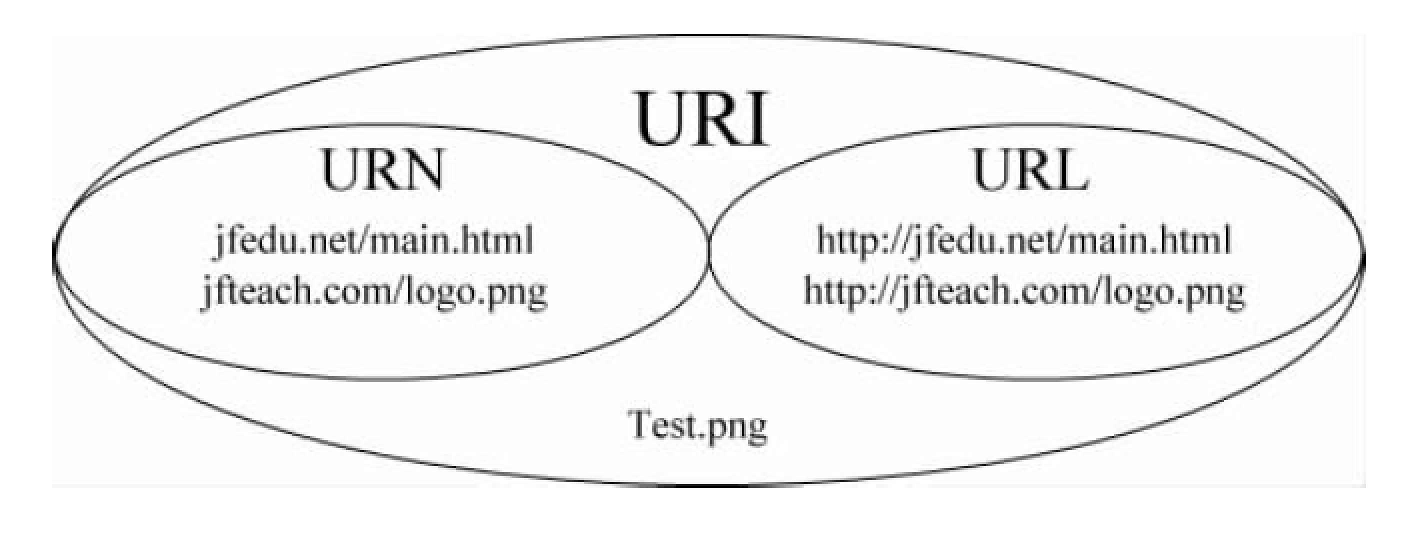


图 9-2　URI、URN、URL 关联与区别

三种资源标识，其中 URL 资源标识方式使用最为广泛，完整的 URL 标识格式如下：

```
protocol://host[:port]/path/.../[?query-string][#anchor]
```

参数详解如下：

- protocol：基于某种协议，常见协议有 HTTP、HTTPS、FTP、RSYNC 等。
- host：服务器的 IP 地址或者域名。
- port：服务器的端口号，如果是 HTTP 80 端口，默认可以省略。
- path：访问资源在服务器的路径。
- query-string：传递给服务器的参数及字符串。
- anchor-：锚定结束。

HTTP URL 案例演示如下：

http://www.jfedu.net/newindex/plus/list.php? tid=2#jfedu

参数对应关系如下：

- protocol：HTTP 协议。
- host：www.jfedu.net。
- path：/newindex/plus/list.php。
- query-string：tid=2。
- anchor：jfedu。

9.3 HTTP 与端口通信

HTTP Web 服务器默认在本机会监听 80 端口，不仅 HTTP 会开启监听端口，其实每个软件程序在 Linux 系统中运行，会以进程的方式启动，程序就会启动并监听本地接口的端口，那么为什么会引入端口这个概念呢？

端口是 TCP/IP 协议中应用层进程与传输层协议实体间的通信接口，端口是操作系统可分配的一种资源，应用程序通过系统调用与某个端口绑定后，传输层传给该端口的数据会被该进程接收，相应进程发给传输层的数据都通过该端口输出。

在网络通信过程中，需要唯一识别通信两端设备的端点，就是使用端口识别运行于某主机中的应用程序。如果没有引入端口，则只能通过 PID 进程号进行识别，而 PID 进程号是系统动态分配的，不同的系统会使用不同的进程标识符，应用程序在运行之前没有明确的进程号，如果需要运行后再广播进程号则很难保证通信的顺利进行。

而引入端口后，就可以利用端口号识别应用程序，同时通过固定端口号来识别和使用某些公共服务，例如 HTTP 默认使用 80 端口，而 FTP 使用 21、20 端口，MySQL 则使用 3306 端口。

使用端口还有一个原因就是随着计算机网络技术的发展，物理机器上的硬件接口已不能满足网络通信的要求，而 TCP/IP 协议模型作为网络通信的标准就解决了这个通信难题。

TCP/IP 协议中引入了一种被称为套接字(socket)的应用程序接口。基于 socket 接口技术，一台计算机就可以与任何一台具有 socket 接口的计算机进行通信，而监听的端口在服务器端也称之为 socket 接口。

9.4 HTTP request 与 response 详解

客户端浏览器向 Web 服务器发起 request，Web 服务器接到 request 后进行处理，会生成相应的 response 信息返给浏览器，客户端浏览器收到服务器返回的 response 信息，会对信息进行解析处理，最终用户看到浏览器展示 Web 服务器的网页内容。

客户端发起 request，request 消息分为 3 个部分，分别包括 request line、request header、body，如图 9-3 所示。



图 9-3　HTTP request message 组成

UNIX/Linux 系统中执行 curl -v 命令可以打印访问 Web 服务器的 request 及 response 详细处理流程，命令如下，流程如图 9-4 所示。

```
curl  -v http://192.168.111.131/index.html
```

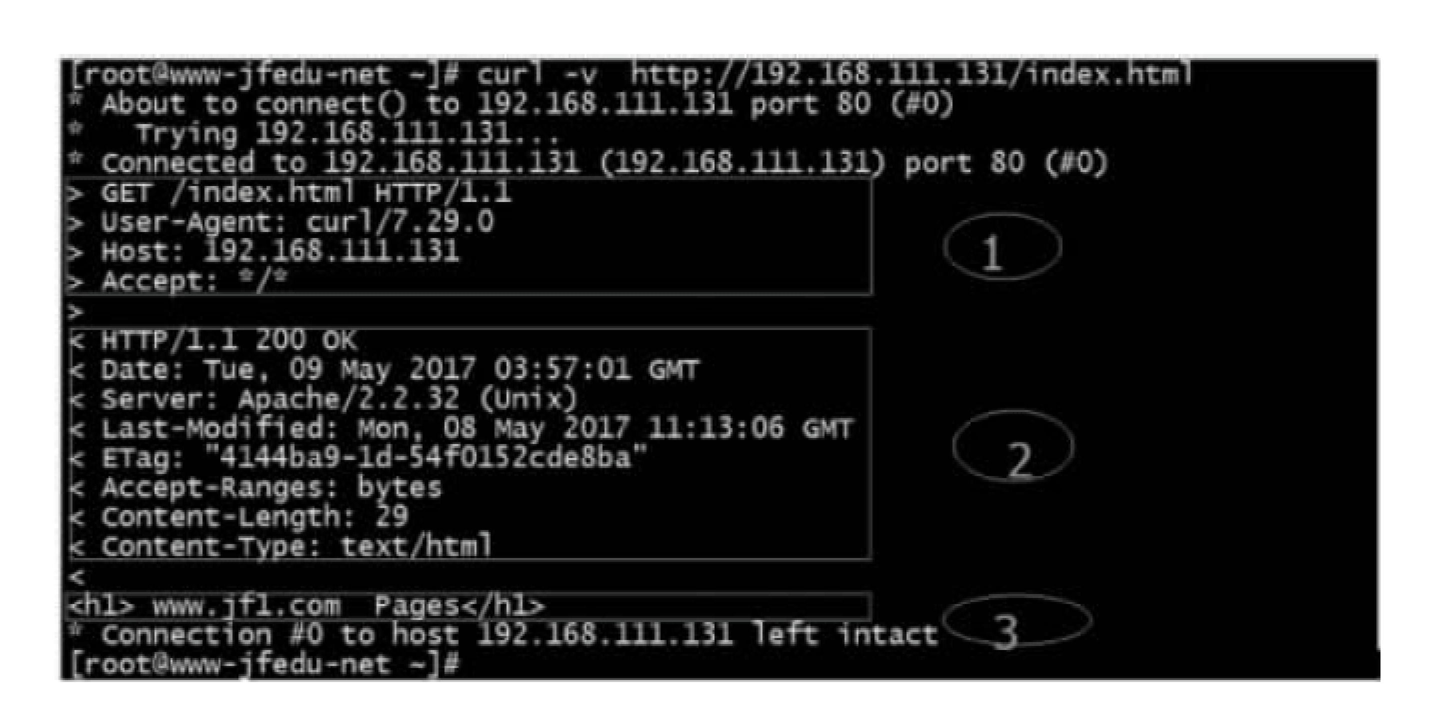


图 9-4　request 及 response 请求回应流程

（1）request 信息详解如表 9-1 所示。

表 9-1　request 请求头详解

GET/index.html HTTP/1.1	请　求　行	request message
User-Agent：curl/7.29.0 Host：192.168.111.131 Accept：*/* …	请求头部	
>	空行	
>	请求 body	

说明：

- 第一部分：请求行，指定请求类型、访问的资源及使用的 HTTP 协议版本。GET 表示 request 请求类型为 GET；/index.html 表示访问的资源；HTTP/1.1 表示协议版本。
- 第二部分：请求头部，请求行下一行起，指定服务器要使用的附加信息。User-Agent 表示用户使用的代理软件，常指浏览器；HOST 表示请求的目的主机。
- 第三部分：空行，请求头部后面的空行表示请求头发送完毕。
- 第四部分：请求数据也叫 body，可以添加任意的数据，Get 请求的 body 内容默认为空。

（2）response 信息详解如表 9-2 所示。

表 9-2　response 请求头详解

HTTP/1.1 200 OK	响应行	response message
Server：Apache/2.2.32 Date：Tue，09 May 2017 Content-Type：text/html …	响应头部	
>	空行	
< h1 > www.jf1.com Pages </h1 >	响应 body	

说明：

- 第一部分：响应状态行，包括 HTTP 协议版本号、状态码、状态消息。HTTP/1.1 表示 HTTP 协议版本号；200 表示返回状态码；OK 表示状态消息。
- 第二部分：消息报头，响应头部附加信息。Date 表示生成响应的日期和时间；Content-Type 表示指定 MIME 类型的 HTML(text/html)，编码类型是 UTF-8，记录文件资源的 Last-Modified 时间。
- 第三部分：空行，表示消息报头响应完毕。
- 第四部分：响应正文，服务器返回给客户端的文本信息。

（3）request 请求方法根据请求的资源不同，有如下请求方法。

- GET 方法：向特定的资源发出请求，获取服务器端数据。
- POST 方法：向 Web 服务器提交数据进行处理请求，常指提交新数据。
- PUT 方法：向 Web 服务器提交上传最新内容，常指更新数据。
- DELETE 方法：请求删除 request-URL 所标识的服务器资源。
- TRACE 方法：回显服务器收到的请求，主要用于测试或诊断。
- CONNECT 方法：HTTP/1.1 协议中预留给能够将连接改为管道方式的代理服务器。
- OPTIONS 方法：返回服务器针对特定资源所支持的 HTTP 请求方法。
- HEAD 方法：HEAD 方法跟 GET 方法相同，只不过服务器响应时不会返回消息体。

9.5　HTTP 1.0/1.1 协议区别

HTTP 协议定义服务器端和客户端之间文件传输的沟通方式，HTTP 1.0 运行方式，如图 9-5 所示。

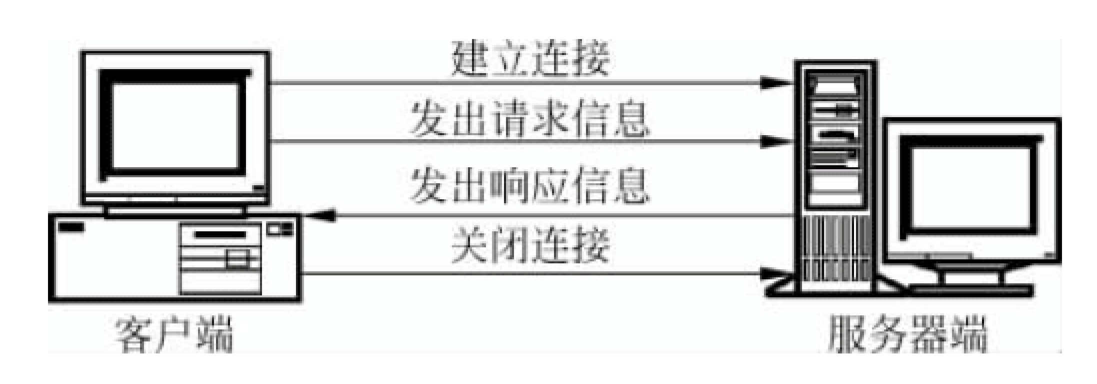


图 9-5　HTTP 1.0 客户端、服务器传输模式

说明如下：

- 基于 HTTP 协议的客户/服务器模式的信息交换过程，如图 9-5 所示，它分为 4 个过程，即建立连接、发送请求信息、发送响应信息、关闭连接。
- 浏览器与 Web 服务器的连接过程是短暂的，每次连接只处理一个请求和响应。对每一个页面的访问，浏览器与 Web 服务器都要建立一次单独的连接。
- 浏览器到 Web 服务器之间的所有通信都是完全独立分开的请求和响应。

HTTP 1.1 运行方式，如图 9-6 所示。

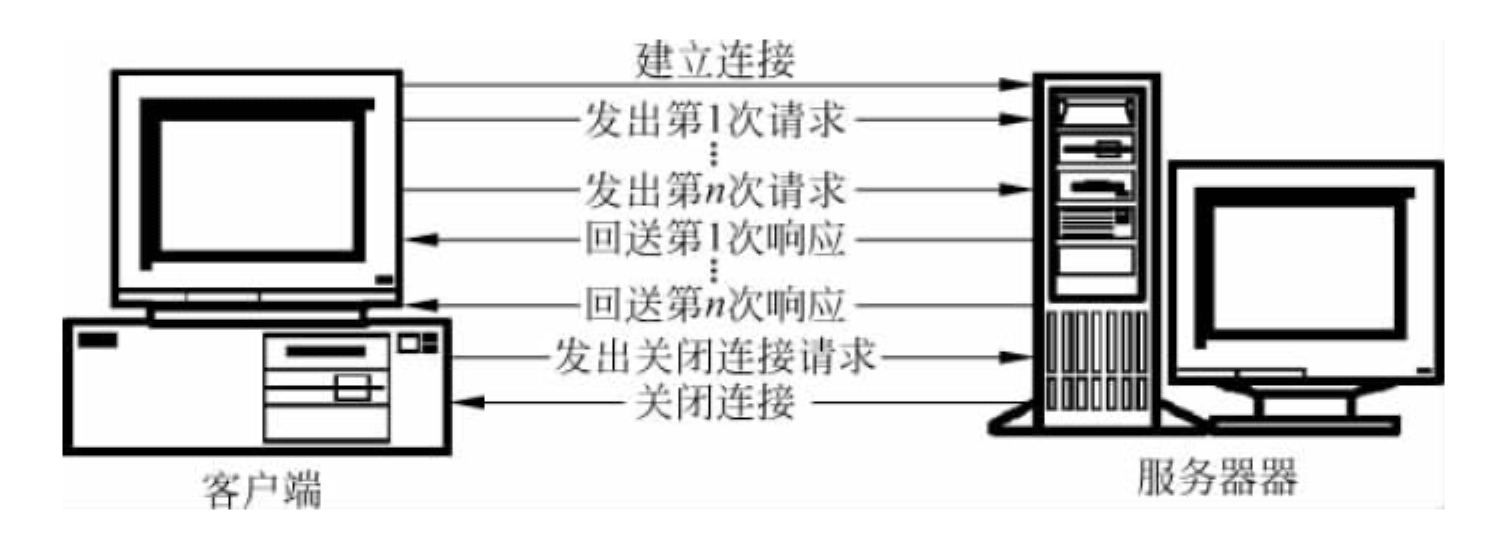


图 9-6　HTTP 1.1 客户端、服务器传输模式

说明如下：

- 在一个 TCP 连接上可以传送多个 HTTP 请求和响应；
- 多个请求和响应过程可以重叠；
- 增加了更多的请求头和响应头，比如 Host、If-Unmodified-Since 请求头等。

9.6　HTTP 状态码详解

HTTP 状态码(HTTP status code)是用来表示 Web 服务器 HTTP response 状态的 3 位数字代码，常见的状态码范围分类如下：

- 100～199：用于指定客户端相应的某些动作。
- 200～299：用于表示请求成功。
- 300～399：已移动的文件且被包含在定位头信息中指定新的地址信息。
- 400～499：用于指出客户端的错误。
- 500～599：用于支持服务器错误。

HTTP 协议 response 常用状态码详解如表 9-3 所示。

表 9-3　HTTP 常用状态码

HTTP 状态码	状态码英文含义	状态码中文含义
100	continue	HTTP/1.1 新增状态码，表示继续，客户端继续请求 HTTP 服务器
101	switching protocols	服务器根据客户端的请求切换协议，切换到 HTTP 的新版本协议

续表

HTTP 状态码	状态码英文含义	状态码中文含义
200	ok	HTTP 请求完成，常用于 GET、POST 请求中
301	moved permanently	永久移动，请求的资源已被永久的移动到新 URI
302	found	临时移动，资源临时被移动，客户端应继续使用原有 URI
304	not modified	文件未修改，请求的资源未修改，服务器返回此状态码时，常用于缓存
400	bad request	客户端请求的语法错误，服务器无法解析或者访问
401	unauthorized	请求要求用户的身份认证
402	payment required	此状态码保留，为以后使用
403	forbidden	服务器理解请求客户端的请求，但是拒绝执行此请求
404	not found	服务器没有该资源，请求的文件找不到
405	method not allowed	客户端请求中的方法被禁止
406	not acceptable	服务器无法根据客户端请求的内容特性完成请求
499	client has closed connection	服务器端处理的时间过长
500	internal server error	服务器内部错误，无法完成请求
502	bad gateway	服务器返回错误代码或者代理服务器错误的网关
503	service unavailable	服务器无法响应客户端请求，或者后端服务器异常
504	gateway time-out	网关超时或者代理服务器超时
505	HTTP version not supported	服务器不支持请求的 HTTP 协议的版本，无法完成处理

9.7 HTTP MIME 类型支持

浏览器接收到 Web 服务器的 response 信息，浏览器会进行解析，在解析页面之前，浏览器必须启动本地相应的应用程序来处理获取到的文件类型。

基于多用途互联网邮件扩展类型(multipurpose internet mail extensions，MIME)，可以明确某种文件在客户端用某种应用程序来打开，当该扩展名文件被访问的时候，浏览器会自动使用指定应用程序来打开，设计之初是为了在发送电子邮件时附加多媒体数据，让邮件客户程序能根据其类型进行处理。然而当它被 HTTP 协议支持后，它使得 HTTP 传输的不仅是普通的文本，可以支持更多文件类型、多媒体音、视频等。

在 HTTP Response 消息中，MIME 类型被定义在 Content-Type header 中，例如 Content-Type：text/html，表示默认指定该文件为 HTML 类型，在浏览器端会以 HTML 格式来处理。

在最早的 HTTP 协议中，并没有附加的数据类型信息，所有传送的数据都被客户程序解释为超文本标记语言 HTML 文档，为了支持多媒体数据类型，新版 HTTP 协议中就使用了附加在文档之前的 MIME 数据类型信息来标识数据类型，如表 9-4 所示。

表 9-4 HTTP MIME 类型详解

MIME types(MIME 类型)	dateiendung (扩展名)	bedeutung(备注)
application/msexcel	*.xls *.xla	Microsoft Excel Dateien
application/mshelp	*.hlp *.chm	Microsoft Windows Hilfe Dateien
application/mspowerpoint	*.ppt *.ppz *.pps *.pot	Microsoft Powerpoint Dateien
application/msword	*.doc *.dot	Microsoft Word Dateien
application/octet-stream	*.exe	exe
application/pdf	*.pdf	Adobe PDF-Dateien
application/post******	*.ai *.eps *.ps	Adobe Post****** -Dateien
application/rtf	*.rtf	Microsoft RTF-Dateien
application/x-httpd-php	*.php *.phtml	PHP-Dateien
application/x-java******	*.js	serverseitige Java****** -Dateien
application/x-shockwave-flash	*.swf *.cab	Flash Shockwave-Dateien
application/zip	*.zip	ZIP-Archivdateien
audio/basic	*.au *.snd	Sound-Dateien
audio/mpeg	*.mp3	MPEG-Dateien
audio/x-midi	*.mid *.midi	MIDI-Dateien
audio/x-mpeg	*.mp2	MPEG-Dateien
audio/x-wav	*.wav	Wav-Dateien
image/gif	*.gif	GIF-Dateien
image/jpeg	*.jpeg *.jpg *.jpe	JPEG-Dateien
image/x-windowdump	*.xwd	X-Windows Dump
text/css	*.css	CSS Stylesheet-Dateien
text/html	*.htm *.html *.shtml	-Dateien
text/java******	*.js	Java****** -Dateien
text/plain	*.txt	reine Textdateien
video/mpeg	*.mpeg *.mpg *.mpe	MPEG-Dateien
video/vnd.rn-realvideo	*.rmvb	realplay-Dateien
video/quicktime	*.qt *.mov	Quicktime-Dateien
video/vnd.vivo	* viv *.vivo	Vivo-Dateien

第 10 章



Apache Web 服务器企业实战

万维网(world wide web,WWW)服务器,也称之为 Web 服务器,主要功能是提供网上信息浏览服务。WWW 是 Internet 的多媒体信息查询工具,是 Internet 上飞快发展的服务,也是目前应用最广泛的服务。正是因为有了 WWW 软件,才使得近年来 Internet 迅速发展。

目前主流的 Web 服务器软件包括 Apache、Nginx、Lighttpd、IIS、Resin、Tomcat、WebLogic、Jetty 等。

本章向读者介绍 Apache Web 服务器发展历史、Apache 工作模式深入剖析、Apache 虚拟主机、配置文件详解及 Apache rewrite 企业实战等内容。

10.1　Apache Web 服务器入门简介

Apache HTTP server 是 Apache 软件基金会的一个开源的网页服务器,可以运行在几乎所有广泛使用的计算机平台上,由于其跨平台和安全性被广泛使用,是目前最流行的 Web 服务器端软件之一。

Apache 服务器是一个多模块化的服务器,经过多次修改,成为目前世界使用排名第一的 Web 服务器软件。Apache 取自"a patchy server"的读音,即充满补丁的服务器,因为 Apache 基于 GPL 发布,大量开发者不断为 Apache 贡献新的代码、功能、新的特性、修改原来的缺陷。

Apache 服务器的特点是使用简单、速度快、性能稳定,可以作为负载均衡及代理服务器来使用。

10.2　Prefork MPM 工作原理

每辆汽车都有发动机引擎,不同的引擎,对车子运行效率也不一样,同样 Apache 也有类似工作引擎或者处理请求的模块,称之为多路处理模块(multi-processing modules,

MPM)，Apache Web 服务器有三种处理模块：Prefork MPM、Worker MPM、Event MPM。

在企业中 Apache 最常用的处理模块为 Prefork MPM 和 Worker MPM。Event MPM 不支持 HTTPS 方式，官网也给出“This MPM is experimental, so it may or may not work as expected”的提示，所以 Event MPM 很少被使用。

默认 Apache 处理模块为 Prefork MPM 方式，Prefork 采用的预派生子进程方式，Prefork 用单独的子进程来处理不同的请求，进程之间是彼此独立的，所以比较稳定。

Prefork MPM 的工作原理：控制进程 Master 在最初建立“StartServers”个进程后，为了满足 MinSpareServers 设置的最小空闲进程，所以需创建第一个空闲进程，等待一秒钟，继续创建两个，再等待一秒钟，继续创建四个，依次按照递增指数级创建进程数，最多每秒同时创建 32 个空闲进程，直到满足至少有 MinSpareServers 设置的值为止。

Apache 的预派生模式(Prefork)，基于预派生模式不必在请求到来时再产生新的进程，从而减小了系统开销以增加性能，不过由于 Prefork MPM 引擎是基于多进程方式提供对外服务，每个进程占内存也相对较高。

10.3 Worker MPM 工作原理

相对于 Prefork MPM，Worker 方式是 2.0 版中全新的支持多线程和多进程混合模型的 MPM，由于使用线程来处理，所以可以处理海量的 HTTP 请求，而系统资源的开销要小于基于 Prefork 多进程的方式。Worker 也是基于多进程，但每个进程又生成多个线程，这样可以保证多线程可以获得进程的稳定性。

Worker MPM 的工作原理：控制进程 Master 在最初建立“StartServers”个进程，每个进程会创建 ThreadsPerChild 设置的线程数，多个线程共享该进程内存空间，同时每个线程独立地处理用户的 HTTP 请求。为了不在请求到来时再生成线程，Worker MPM 也可以设置最大最小空闲线程。

Worker MPM 模式下同时处理的请求总数=进程总数 * ThreadsPerChild，也即等于 MaxClients。如果服务器负载很高，当前进程数不满足需求，Master 控制进程会 fork 新的进程，最大进程数不能超过 ServerLimit 数，如果需调整的 StartServers 进程数，需同时调整 ServerLimit 值。

Prefork MPM 与 Worker MPM 引擎区别总结如下：

- Prefork MPM 模式：使用多个进程，每个进程只有一个线程，每个进程在某个确定的时间只能维持一个连接，优点是稳定，但内存开销较高。
- Worker MPM 模式：使用多个进程，每个进程包含多个线程，每个线程在某个确定的时间只能维持一个连接，内存占用量比较小，适合大并发、高流量的 Web 服务器。Worker MPM 缺点是一个线程崩溃，整个进程就会连同其任何线程一起挂掉。

10.4　Apache Web 服务器安装

从 Apache 官方分站点下载目前稳定版本 httpd-2.2.32 版本，目前最新版本为 2.4 版本，下载地址如下：http://mirrors.hust.edu.cn/apache/httpd/httpd-2.2.32.tar.bz2。

Apache Web 服务器安装步骤详解如下：

- tar -xjvf httpd-2.2.32.tar.bz2：tar 工具解压 httpd 包。
- cd httpd-2.2.32/：进入解压后目录。
- yum install apr apr-devel apr-util apr-util-devel -y：安装 apr 相关移植库模块。
- ./configure --prefix=/usr/local/apache2/ --enable-rewrite --enable-so：预编译 Apache，启用 rewrite 规则、启用动态加载库。
- make：编译。
- make install：安装。

Apache2.2.32 安装完毕，如图 10-1 所示。

```
Making install in support
make[1]: 进入目录"/root/httpd-2.2.32/support"
make[2]: 进入目录"/root/httpd-2.2.32/support"
make[2]: 离开目录"/root/httpd-2.2.32/support"
make[1]: 离开目录"/root/httpd-2.2.32/support"
make[1]: 进入目录"/root/httpd-2.2.32"
Installing configuration files
[PRESERVING EXISTING HTDOCS SUBDIR: /usr/local/apache2//ht
[PRESERVING EXISTING ERROR SUBDIR: /usr/local/apache2//err
[PRESERVING EXISTING ICONS SUBDIR: /usr/local/apache2//ico
[PRESERVING EXISTING CGI SUBDIR: /usr/local/apache2//cgi-b
Installing header files
Installing build system files
Installing man pages and online manual
```

图 10-1　Apache2.2.32 安装图解

启动 Apache 服务，临时关闭 SELinux、firewalld 防火墙，命令如下：

```
/usr/local/apache2/bin/apachectl start
setenforce 0
systemctl stop firewalld.service
```

查看 Apache 服务进程，通过客户端浏览器访问 http://192.168.111.131/，如图 10-2 所示。

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# /usr/local/apache2/bin/apachectl start
httpd: Could not reliably determine the server's fully qualified domai
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ps -ef |grep httpd
root     14399     1  0 05:05 ?        00:00:00 /usr/local/apache2//bi
daemon   14400 14399  0 05:05 ?        00:00:00 /usr/local/apache2//bi
daemon   14401 14399  0 05:05 ?        00:00:00 /usr/local/apache2//bi
daemon   14402 14399  0 05:05 ?        00:00:00 /usr/local/apache2//bi
daemon   14403 14399  0 05:05 ?        00:00:00 /usr/local/apache2//bi
daemon   14404 14399  0 05:05 ?        00:00:00 /usr/local/apache2//bi
root     14406  2116  0 05:05 pts/0    00:00:00 grep --color=auto http
[root@www-jfedu-net ~]#
```

(a) Apache启动及查看进程

图 10-2　查看 Apache Web 服务器



(b) 浏览器访问Apache Web服务器

图 10-2 （续）

10.5 Apache 虚拟主机企业应用

企业真实环境中，一台 Web 服务器发布单个网站非常浪费资源，所以一台 Web 服务器上会发布多个网站，少则 3～5 个，多则 2～30 个网站。在一台服务器上发布多网站，也称之为部署多个虚拟主机，Web 虚拟主机配置方法有以下三种：

- 基于单 IP 多个 socket 端口；
- 基于多 IP 地址一个端口；
- 基于单 IP 一个端口不同域名。

其中基于同一端口不同域名的方式在企业中得到广泛应用，以下为基于一个端口不同域名，在一台 Apache Web 服务器上部署多个网站，步骤如下：

（1）创建虚拟主机配置文件 httpd-vhosts.conf，该文件默认已存在，只需去掉 httpd.conf 配置文件中的＃号即可，如图 10-3 所示。

```
# User home directories
#Include conf/extra/httpd-userdir.conf

# Real-time info on requests and configuration
#Include conf/extra/httpd-info.conf

# Virtual hosts
Include conf/extra/httpd-vhosts.conf

# Local access to the Apache HTTP Server Manual
#Include conf/extra/httpd-manual.conf

# Distributed authoring and versioning (WebDAV)
#Include conf/extra/httpd-dav.conf
```

图 10-3 httpd.conf 配置文件开启虚拟主机

（2）修改配置文件/usr/local/apache2/conf/extra/httpd-vhosts.conf 中代码，设置如下：

```
NameVirtualHost *:80
<VirtualHost *:80>
    ServerAdmin support@jfedu.net
    DocumentRoot "/usr/local/apache2/htdocs/jf1"
    ServerName www.jf1.com
    ErrorLog "logs/www.jf1.com_error_log"
    CustomLog "logs/www.jf1.com_access_log" common
```

```
</VirtualHost>
<VirtualHost *:80>
    ServerAdmin support@jfedu.net
    DocumentRoot "/usr/local/apache2/htdocs/jf2"
    ServerName www.jf2.com
    ErrorLog "logs/www.jf2.com_error_log"
    CustomLog "logs/www.jf2.com_access_log" common
</VirtualHost>
```

httpd-vhosts.conf 参数详解如下：

- NameVirtualHost *:80：开启虚拟主机，并且监听本地所有网卡接口的 80 端口。
- <VirtualHost *:80>：虚拟主机配置起始。
- ServerAdmin support@jfedu.net：管理员邮箱。
- DocumentRoot "/usr/local/apache2/htdocs/jf1"：虚拟主机发布目录。
- ServerName www.jf1.com：虚拟主机完整域名。
- ErrorLog "logs/www.jf1.com_error_log"：错误日志路径及文件名。
- CustomLog "logs/www.jf1.com_access_log" common：访问日志路径及文件名。
- </VirtualHost>：虚拟主机配置结束。

(3) 创建 www.jf1.com 及 www.jf2.com 发布目录，重启 Apache 服务，并分别创建 index.html 页面，命令如下：

```
mkdir -p /usr/local/apache2/htdocs/{jf1,jf2}/
/usr/local/apache2/bin/apachectl restart
echo "<h1> www.jf1.com Pages </h1>" >/usr/local/apache2/htdocs/jf1/index.html
echo "<h1> www.jf2.com Pages </h1>" >/usr/local/apache2/htdocs/jf2/index.html
```

(4) Windows 客户端设置 hosts 映射，将 www.jf1.com、www.jf2.com 与 192.168.111.131 IP 进行映射绑定，映射的目的将域名跟 IP 进行绑定，在浏览器可以输入域名，不需要输入 IP 地址，绑定方法是在“C:\Windows\System32\drivers\etc”文件夹中，使用记事本编辑 hosts 文件，加入如下代码，如图 10-4 所示。

```
192.168.111.131 www.jf1.com
192.168.111.131 www.jf2.com
```



图 10-4　Windows 主机 hosts 配置

（5）浏览器访问 www.jf1.com、www.jf2.com，如图 10-5 所示，至此 Apache 基于多域名虚拟主机配置完毕，如果还需添加虚拟主机，直接复制其中一个虚拟主机配置、修改 Web 发布目录即可。



(a) www.jf1.com网站返回内容

(b) www.jf2.com网站返回内容

图 10-5　网站返回内容

10.6　Apache 常用目录学习

Apache 可以基于源码安装、YUM 安装，不同的安装方法，所属的路径特不同，以下为 Apache 常用路径的功能用途：

- /usr/lib64/httpd/modules/：Apache 模块存放路径。
- /var/www/html/：YUM 安装 Apache 网站发布目录。
- /var/www/error/：服务器设置错误信息，浏览器显示。
- var/www/icons/：Apache 小图标文件存放目录。
- var/www/cgi-bin/：可执行的 CGI 程序存放目录。
- /var/log/httpd/：Apache 日志目录。
- /usr/sbin/apachectl：Apache 启动脚本。
- /usr/sbin/httpd：Apache 二进制执行文件。
- /usr/bin/htpasswd：设置 Apache 目录密码访问。
- /usr/local/apache2/bin：Apache 命令目录。
- /usr/local/apache2/build：Apache 构建编译目录。
- /usr/local/apache2/htdocs/：源码安装 Apache 网站发布目录。
- /usr/local/apache2/cgi-bin：可执行的 CGI 程序存放目录。

- /usr/local/apache2/include：Apache 引用配置文件目录。
- /usr/local/apache2/logs：Apache 日志目录。
- /usr/local/apache2/man：Apache 帮助文档目录。
- /usr/local/apache2/manual：Apache 手册。
- /usr/local/apache2/modules：Apache 模块路径。

10.7 Apache 配置文件详解

Apache 的配置文件是 Apache Web 难点，读者需要掌握配置文件中每个参数的含义，才能理解并在日常运维中去解决 Apache 遇到的故障，以下为 Apache 配置文件详解：

- ServerTokens OS：显示服务器的版本和操作系统内核版本。
- ServerRoot "/usr/local/apache2/"：Apache 主配置目录。
- PidFile run/httpd.pid：PidFile 进程文件。
- Timeout 60：不论接收或发送，当持续连接等待超过 60 秒则该次连接就中断。
- KeepAlive Off：关闭持续性的连接。
- MaxKeepAliveRequests 100：当 KeepAlive 设置为 On 的时候，该数值可以决定此次连接能够传输的最大传输数量。
- KeepAliveTimeout 65：当 KeepAlive 设置为 On 的时候，该连接在最后一次传输后等待延迟的秒数。
- <IfModule prefork.c>：Prefork MPM 引擎配置段。
- StartServers 8：默认启动 Apache 工作进程数。
- MinSpareServers 5：最小空闲进程数。
- MaxSpareServers 20：最大空闲进程数。
- ServerLimit 4096：Apache 服务器最多进程数。
- MaxClients 4096：每秒支持的最大客户端并发。
- MaxRequestsPerChild 4000：每个进程能处理的最大请求数。
- </IfModule>：定义模块，模块标签。
- <IfModule worker.c>：Worker MPM 引擎配置段。
- StartServers 8：默认启动 Apache 工作进程数。
- MaxClients 4000：每秒支持的最大客户端并发。
- MinSpareThreads 25：最小空闲线程数。
- MaxSpareThreads 75：最大空闲线程数。
- ThreadsPerChild 75：每个进程启动的线程数。
- MaxRequestsPerChild 0：每个进程能处理的最大请求数，0 表示无限制。
- </IfModule>：定义模块，模块标签。
- LoadModule mod_version.so：静态加载 Apache 相关模块。

- ServerAdmin support@jfedu.net：管理员邮箱，网站异常，错误信息会发生至该邮箱。
- DocumentRoot "/usr/local/apache2/htdocs/"：Apache 网站默认发布目录。
- <Directory "/data/webapps/www1">：设置/data/webapps/www1 目录权限。
- AllowOverride All：加载发布目录中.htaccess 文件。
- Options -Indexes FollowSymLinks：禁止发布目录以目录方式被浏览。
- Order allow,deny：访问顺序，先检查允许设置，没有允许的设置则全部拒绝。
- Allow from all：允许所有客户端访问。
- </Directory>：定义目录，目录标签。
- AllowOverride：设置为 None 时，目录中.htaccess 文件将被完全忽略，当指令设置为 All 时，.htaccess 文件生效。
- Options -Indexes FollowSymLinks：禁止浏览目录，去掉"-"，表示浏览目录，常用于下载站点。
- Order allow,deny：默认情况下禁止所有客户机访问。
- Order deny,allow：默认情况下允许所有客户机访问。
- Allow from all：允许所有客户机访问。

10.8 Apache rewrite 规则实战

rewrite 规则也称为规则重写，主要功能是实现浏览器访问 HTTP URL 的跳转，其规则表达式是基于 Perl 语言。通常而言，几乎所有的 Web 服务器均可以支持 URL 重写。rewrite URL 规则重写的用途如下：

- 对搜索引擎优化(search engine optimization,SEO)友好，利于搜索引擎抓取网站页面；
- 隐藏网站 URL 真实地址，浏览器显示更加美观；
- 网站变更升级，可以基于 rewrite 临时重定向到其他页面。

Apache Web 服务器如需要使用 rewrite 功能，须添加 rewrite 模块，基于源码安装是指定参数"--enable-rewrite"。还有一种方法可以动态添加模块，以 DSO 模式安装 Apache，利用模块源码和 Apache apxs 工具完成 rewrite 模块的添加。

使用 Apache rewrite，除了安装 rewrite 模块之外，还需在 httpd.conf 中的全局配置段或者虚拟主机配置段设置如下指令来开启 rewrite 功能：

```
RewriteEngine on
```

Apache rewrite 规则使用中有三个概念需要理解，分别为：rewrite 结尾标识符、rewrite 规则常用表达式、Apache rewrite 变量，以下为三个概念的详解：

(1) Apache rewrite 结尾标识符，用于 rewrite 规则末尾，表示规则的执行属性。详解

如下：

- R[=code](force redirect)：强制外部重定向。
- G(force URL to be gone)：强制 URL 为 gone，返回 410HTTP 状态码。
- P(force proxy)：强制使用代理转发。
- L(last rule)：匹配当前规则为最后一条匹配规则，停止匹配后续规则。
- N(next round)：重新从第一条规则开始匹配。
- C(chained with next rule)：与下一条规则关联。
- T=MIME-type(force MIME type)：强制 MIME 类型。
- NC(no case)：不区分大小写。

(2) Apache rewrite 规则常用表达式，主要用于匹配参数、字符串及过滤设置。详解如下：

- .：匹配任何单字符。
- [word]：匹配字符串 word。
- [^word]：不匹配字符串 word。
- jfedu|jfteach：可选择字符串 jfedu|jfteach。
- ?：匹配 0 到 1 个字符。
- *：匹配 0 到多个字符。
- +：匹配 1 到多个字符。
- ^：字符串开始标志。
- $：字符串结束标志。
- \n：转义符标志。

(3) Apache rewrite 变量，常用于匹配 HTTP 请求头信息、浏览器主机名、URL 等。代码如下：

```
HTTP headers:HTTP_USER_AGENT, HTTP_REFERER, HTTP_COOKIE, HTTP_HOST, HTTP_ACCEPT;
connection & request: REMOTE_ADDR, QUERY_STRING;
server internals: DOCUMENT_ROOT, SERVER_PORT, SERVER_PROTOCOL;
system stuff: TIME_YEAR, TIME_MON, TIME_DAY。
```

详解如下：

- HTTP_USER_AGENT：用户使用的代理，例如浏览器。
- HTTP_REFERER：告知服务器，从哪个页面来访问的。
- HTTP_COOKIE：客户端缓存，主要用于存储用户名和密码等信息。
- HTTP_HOST：匹配服务器 ServerName 域名。
- HTTP_ACCEPT：客户端的浏览器支持的 MIME 类型。
- REMOTE_ADDR：客户端的 IP 地址。
- QUERY_STRING：URL 中访问的字符串。
- DOCUMENT_ROOT：服务器发布目录。

- SERVER_PORT：服务器端口。
- SERVER_PROTOCOL：服务器端协议。
- TIME_YEAR：年。
- TIME_MON：月。
- TIME_DAY：日。

(4) rewrite 规则实战案例，以下配置均配置在 httpd.conf 或者 vhosts.conf 中，企业中常用的 rewrite 案例具体步骤如下：

- 将 jfedu.net 跳转至 www.jfedu.net，说明如下：
- RewriteEngine on：启用 rewrite 引擎。
- RewriteCond %{HTTP_HOST} ^jfedu.net [NC]：匹配以 jfedu.net 开头的域名，NC 忽略大小写。
- RewriteRule ^/(.*)$http://www.jfedu.net/$1 [L]：(.*)表示任意字符串，$1 表示引用(.*)的中任意内容。
- 将 www.jf1.com、www.jf2.com、jfedu.net 跳转至 www.jfedu.net，OR 含义表示或者，代码如下：

```
RewriteEngine on
RewriteCond %{HTTP_HOST} www.jf1.com          [NC,OR]
RewriteCond %{HTTP_HOST} www.jf2.com          [NC,OR]
RewriteCond %{HTTP_HOST} ^jfedu.net           [NC]
RewriteRule ^/(.*)$ http://www.jfedu.net/$1   [L]
```

- 访问 www.jfedu.net 首页，跳转至 www.jfedu.net/newindex/，R=301 表示永久重定向，代码如下：

```
RewriteEngine on
RewriteRule ^/$ http://www.jfedu.net/newindex/ [L,R=301]
```

- 访问/newindex/plus/view.php?aid=71 跳转至 http://www.jfedu.net/linux/，代码如下：

```
RewriteEngine on
RewriteCond %{QUERY_STRING} ^tid=(.+)$                [NC]
RewriteRule ^/forum\.php$ /jfedu/thread-new-%1.html? [R=301,L]
```

- 访问 www.jfedu.net 首页，内容访问 www.jfedu.net/newindex/，但是浏览器 URL 地址不改变，代码如下：

```
RewriteEngine on
RewriteCond %{HTTP_HOST} ^www.jfedu.net [NC]
RewriteRule ^/$ /newindex/              [L]
```

- 访问/forum.php?tid=107258 跳转至/jfedu/thread-new-107258.html，代码如下：

```
RewriteEngine on
RewriteCond %{QUERY_STRING} ^tid=(.+)$                 [NC]
RewriteRule ^/forum\.php$ /jfedu/thread-new-%1.html? [R=301,L]
```

- 访问/xxx/123456 跳转至/xxx?id=123456，代码如下：

```
RewriteEngine on
rewriteRule ^/(.+)/(\d+)$ /$1?id=$2 [L,R=301]
```

- 判断是否使用移动端访问网站，移动端访问跳转至 m.jfedu.net，代码如下：

```
RewriteEngine on
RewriteCond %{HTTP_USER_AGENT} ^iPhone              [NC,OR]
RewriteCond %{HTTP_USER_AGENT} ^Android             [NC,OR]
RewriteCond %{HTTP_USER_AGENT} ^WAP                 [NC]
rewriteRule ^/$        http://m.jfedu.net/index.html  [L,R=301]
rewriteRule ^/(.*)/$  http://m.jfedu.net/$1          [L,R=301]
```

- 访问/10690/jfedu/123 跳转至/index.php?tid/10690/items=123，[0-9]表示任意一个数字，+表示多个，(.+)表示任何多个字符，代码如下：

```
RewriteEngine on
RewriteRule ^/([0-9]+)/jfedu/(.+)$ /index.php?tid/$1/items=$2 [L,R=301]
```

第 11 章



MySQL 服务器企业实战

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 旗下公司。MySQL 是当下最流行的关系型数据库管理系统，在 Web 应用方面 MySQL 是最好的关系数据库管理系统（relational database management system，RDBMS）应用软件之一。

本章向读者介绍关系型数据库特点、MySQL 数据库引擎特点、数据库安装配置、SQL 案例操作、数据库索引、慢查询、MySQL 数据库集群实战等内容。

11.1　MySQL 数据库入门简介

MySQL 是一种关联数据库管理系统，关联数据库将数据保存在不同的表格中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。

MySQL 数据库主要用于存储各类信息数据，例如：员工姓名、身份证 ID、商城订单及金额、销售业绩及报告、学生考试成绩、网站帖子、论坛用户信息、系统报表等。

MySQL 软件采用了双授权政策，它分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。由于其社区版的性能卓越，搭配 PHP 和 Apache 可组成良好的开发环境。

RDBMS 是将数据组织为相关的行和列的系统，而管理关系数据库的计算机软件就是关系数据库管理系统，常用的关系型数据库软件有：MySQL、MariaDB、Oracle、SQL Server、PostgreSQL、DB2 等。

RDBMS 数据库的特点如下：

- 数据以表格的形式出现；
- 每行记录数据的真实内容；
- 每列记录数据真实内容的数据域；
- 无数的行和列组成一张表；

- 若干的表组成一个数据库。

目前 Web 主流架构是 LAMP(Linux+Apache+MySQL+PHP),MySQL 更是得到各位 IT 运维、DBA 的青睐,虽然 MySQL 数据库已被 Orcacle 公司收购,不过好消息是原 MySQL 创始人已独立出来重新开发了 MariaDB 数据库,开源免费,目前越来越多的企业开始尝试使用。MariaDB 数据库兼容 MySQL 数据库所有的功能和相关参数。

MySQL 数据库运行在服务器前,需要选择启动的引擎,好比一辆轿车,性能好的发动机会提升轿车的性能,从而启动、运行更加的高效。同样 MySQL 也有类似发动机引擎,这里称之为 MySQL 引擎。

MySQL 引擎包括:ISAM、MyISAM、InnoDB、Memory、CSV、BlackHole、Archive、Performance_Schema、Berkeley、Merge、Federated、Cluster/NDB 等,其中 MyISAM、InnoDB 使用最为广泛,以下为 MyISAM、BDB、Memory、InnoDB 以及 Archive 之间的引擎功能的对比,如表 11-1 所示。

表 11-1 引擎功能对比

引擎特性	MyISAM	BDB	Memory	InnoDB	Archive
批量插入的速度	高	高	高	中	非常高
集群索引	不支持	不支持	不支持	支持	不支持
数据缓存	不支持	不支持	支持	支持	不支持
索引缓存	支持	不支持	支持	支持	不支持
数据可压缩	支持	不支持	不支持	不支持	支持
硬盘空间使用	低	低	null	高	非常低
内存使用	低	低	中等	高	低
外键支持	不支持	不支持	不支持	支持	不支持
存储限制	没有	没有	有	64TB	没有
事务安全	不支持	支持	不支持	支持	不支持
锁机制	表锁	页锁	表锁	行锁	行锁
B 树索引	支持	支持	支持	支持	不支持
哈希索引	不支持	不支持	支持	支持	不支持
全文索引	支持	不支持	不支持	不支持	不支持

性能总结如下:

(1) MyISAM、MySQL 5.0 之前的默认数据库引擎,最为常用。拥有较高的插入、查询速度,但不支持事务。

(2) InnoDB 事务型数据库的首选引擎,支持 ACID 事务,ACID 包括原子性(atomicity)、一致性(consistency)、隔离性(isolation)、持久性(durability),一个支持事务(transaction)的数据库,必须具有这 4 种特性,否则在执行事务过程中无法保证数据的正确性。

（3）MySQL 5.5 之后默认引擎为 InnoDB，InnoDB 支持行级锁定，支持事物、外键等功能。

（4）BDB 源自 Berkeley DB，事务型数据库的另一种选择，支持 Commit 和 Rollback 等其他事务特性。

（5）Memory 所有数据置于内存的存储引擎，拥有极高的插入、更新和查询效率。但是会占用和数据量成正比的内存空间，并且其内容会在 MySQL 重新启动时丢失。

（6）MySQL 常用的两大引擎有 MyISAM 和 InnoDB，那么它们之间的区别是什么，根据不同场合该如何进行选择。MyISAM 类型的数据库表强调的是性能，其执行数度比 InnoDB 类型更快，但不提供事务支持，不支持外键，如果执行大量的 select（查询）操作，MyISAM 是更好的选择，支持表锁。InnoDB 提供事务支持事务、外部键、行级锁等高级数据库功能，执行大量的 insert 或 update 操作，出于性能方面的考虑，可以使用 InnoDB 引擎。

11.2 MySQL 数据库安装方式

MySQL 数据库安装方法有两种：一种是 yum/rpm 通过 YUM 源在线安装；另一种是通过源码软件编译安装。

（1）YUM 方式安装 MySQL 的方法，执行命令详解如下：

- yum install mysql-server mysql-devel mysql-libs -y：CentOS 6.X YUM 安装。
- yum install mariadb-server mariadb mariadb-libs -y：CentOS 7.X YUM 安装。

（2）源码安装 MySQL 5.5.20 方法，通过 cmake、make、make install 三个步骤实现，命令如下：

```
wget http://down1.chinaunix.net/distfiles/mysql-5.5.20.tar.gz
yum install cmake ncurses-devel ncurses -y
cmake . -DCMAKE_INSTALL_PREFIX=/usr/local/mysql55/ \
-DMYSQL_UNIX_ADDR=/tmp/mysql.sock \
-DMYSQL_DATADIR=/data/mysql \
-DSYSCONFDIR=/etc \
-DMYSQL_USER=mysql \
-DMYSQL_TCP_PORT=3306 \
-DWITH_XTRADB_STORAGE_ENGINE=1 \
-DWITH_INNOBASE_STORAGE_ENGINE=1 \
-DWITH_PARTITION_STORAGE_ENGINE=1 \
-DWITH_BLACKHOLE_STORAGE_ENGINE=1 \
-DWITH_MYISAM_STORAGE_ENGINE=1 \
-DWITH_READLINE=1 \
-DENABLED_LOCAL_INFILE=1 \
-DWITH_EXTRA_CHARSETS=1 \
-DDEFAULT_CHARSET=utf8 \
```

```
-DDEFAULT_COLLATION=utf8_general_ci \
-DEXTRA_CHARSETS=all \
-DWITH_BIG_TABLES=1 \
-DWITH_DEBUG=0
make
make install
```

（3）MySQL 源码安装参数详解如下：

- cmake . -DCMAKE_INSTALL_PREFIX=/usr/local/mysql55：cmake 预编译。
- -DMYSQL_UNIX_ADDR=/tmp/mysql.sock：MySQL socket 通信文件位置。
- -DMYSQL_DATADIR=/data/mysql：MySQL 数据存放路径。
- -DSYSCONFDIR=/etc：配置文件路径。
- -DMYSQL_USER=mysql：MySQL 运行用户。
- -DMYSQL_TCP_PORT=3306：MySQL 监听端口。
- -DWITH_XTRADB_STORAGE_ENGINE=1：开启 XtraDB 引擎支持。
- -DWITH_INNOBASE_STORAGE_ENGINE=1：开启 InnoDB 引擎支持。
- -DWITH_PARTITION_STORAGE_ENGINE=1：开启 Partition 引擎支持。
- -DWITH_BLACKHOLE_STORAGE_ENGINE=1：开启 BlackHole 引擎支持。
- -DWITH_MYISAM_STORAGE_ENGINE=1：开启 MyISAM 引擎支持。
- -DWITH_READLINE=1：启用快捷键功能。
- -DENABLED_LOCAL_INFILE=1：允许从本地导入数据。
- -DWITH_EXTRA_CHARSETS=1：支持额外的字符集。
- -DDEFAULT_CHARSET=utf8：默认字符集 UTF-8。
- -DDEFAULT_COLLATION=utf8_general_ci：检验字符。
- -DEXTRA_CHARSETS=all：安装所有扩展字符集。
- -DWITH_BIG_TABLES=1：将临时表存储在磁盘上。
- -DWITH_DEBUG=0：禁止调试模式支持。
- make：编译。
- make install：安装。

（4）将源码安装的 MySQL 数据库服务设置为系统服务，可以使用 chkconfig 管理，并启动 MySQL 数据库，命令如下，详情如图 11-1 所示。

```
cd /usr/local/mysql55/
\cp support-files/my-large.cnf /etc/my.cnf
\cp support-files/mysql.server /etc/init.d/mysqld
chkconfig --add mysqld
chkconfig --level 35 mysqld on
mkdir -p /data/mysql
useradd mysql
/usr/local/mysql55/scripts/mysql_install_db --user=mysql --datadir=/data/mysql/
```

```
-- basedir = /usr/local/mysql55/
ln - s /usr/local/mysql55/bin/ * /usr/bin/
service mysqld restart
```

```
Please report any problems with the /usr/local/mysql55//scripts/mysqlb
[root@localhost mysql55]# service  mysqld  restart
 ERROR! MySQL server PID file could not be found!
Starting MySQL... SUCCESS!
[root@localhost mysql55]# clear
[root@localhost mysql55]#
[root@localhost mysql55]# ps -ef |grep mysql
root      2286     1  0 11:47 pts/0    00:00:00 /bin/sh /usr/local/mys
le=/data/mysql/localhost.pid
mysql     2561  2286  4 11:47 pts/0    00:00:00 /usr/local/mysql55/bin
a/mysql --plugin-dir=/usr/local/mysql55/lib/plugin --user=mysql --log-
sql/localhost.pid --socket=/tmp/mysql.sock --port=3306
root      2585  1234  0 11:48 pts/0    00:00:00 grep mysql
[root@localhost mysql55]#
```

图 11-1 查看 MySQL 启动进程

（5）不设置为系统服务，也可以用源码启动方式，命令如下：

```
cd /usr/local/mysql55
mkdir - p /data/mysql
useradd mysql
/usr/local/mysql55/scripts/mysql_install_db -- user = mysql
 -- datadir = /data/mysql/
 -- basedir = /usr/local/mysql55/
ln - s /usr/local/mysql55/bin/ * /usr/bin/
/usr/local/mysql55/bin/mysqld_safe -- user = mysql &
```

11.3 MySQL 数据库必备命令操作

MySQL 数据库安装完毕之后，对 MySQL 数据库中各种指令的操作变得尤为重要，熟练掌握 MySQL 必备命令是 SA、DBA 必备工作之一。以下为 MySQL 数据库中操作必备命令，所有操作指令均在 MySQL 命令行中操作，不能在 Linux shell 解释器上直接运行。

在 shell 终端执行命令 mysql 或者/usr/local/mysql55/bin/mysql，按 Enter 键，进入 MySQL 命令行界面，如图 11-2 所示。

```
[root@www-jfedu-net ~]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.5.20-log Source distribution

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reser

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input stat

mysql> 
```

图 11-2 MySQL 命令行界面

MySQL 命令行常用命令详解如下，操作结果如图 11-3 所示。

- show databases：查看所有的数据库。
- create database jfedu：创建名为 jfedu 数据库。
- use jfedu：进入 jfedu 数据库。
- show tables：查看数据库里有多少张表。
- create table t1 (id varchar(20),name varchar(20))：创建名为 t1 表，并创建两个字

```
[root@www-jfedu-net ~]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.5.20-log Source distribution

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)

mysql> create database jfedu;
Query OK, 1 row affected (0.00 sec)
```

(a) MySQL命令操作(1)

```
mysql> use jfedu;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql> create table t1 (id varchar(20),name varchar(20));
Query OK, 0 rows affected (0.01 sec)

mysql> insert into t1 values ("1","jfedu");
Query OK, 1 row affected (0.00 sec)

mysql> select * from t1;
+------+-------+
| id   | name  |
+------+-------+
| 1    | jfedu |
+------+-------+
1 row in set (0.00 sec)
```

(b) MySQL命令操作(2)

```
mysql> select * from t1 where id=1 and name='jfedu';
+------+-------+
| id   | name  |
+------+-------+
| 1    | jfedu |
+------+-------+
1 row in set (0.00 sec)

mysql> desc t1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | varchar(20) | YES  |     | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> alter table t1 modify column name varchar(20);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> update t1 set name='jfedu.net' where id=1;
Query OK, 1 row affected (0.01 sec)
```

(c) MySQL命令操作(3)

图 11-3 MySQL 命令操作

段，id、name，varchar 表示设置数据长度，用字符来定义长度单位，其中 1 汉字=2 字符=2 字节。

- insert into t1 values ("1","jfedu")：向表中插入数据。
- select * from t1：查看 t1 表数据内容。
- Select * from t1 where id=1 and age ='jfedu'：id、age 多个条件查询。
- desc t1：查看 t1 表字段内容。
- alter table t1 modify column name varchar(20)：修改 name 字段的长度。
- update t1 set name='jfedu.net' where id=1：修改 name 字段的内容。
- flush privileges：刷新权限。
- delete from t1：清空表内容。
- drop table t1：删除表。
- drop database jfedu：删除 jfedu 数据库。
- show variables like '%char%'：查看数据库字符集。
- show engines：查看 MySQL 存储引擎。
- show variables like'%storage_engine%'：查看 MySQL 默认的存储引擎。
- alter table t1 engine=innodb：修改 MySQL t1 表存储引擎。

11.4 MySQL 数据库字符集设置

计算机中储存的信息都是用二进制数方式来表示的，读者每天看到屏幕显示的英文、汉字等字符也都是二进制数转换之后的结果。通俗地说，将汉字按照某种字符集编码存储在计算机中，称为“编码”。将存储在计算机中的二进制数解析显示出来，称为“解码”，在解码过程中，如果使用了错误的解码规则，会导致显示乱码。

MySQL 数据库在存储数据时，默认编码为 latin1，存储中文字符时，在显示或者 Web 调用时会显示为乱码，为解决该乱码问题，需修改 MySQL 默认字符集为 UTF-8，有以下两种方法：

(1) 编辑 vim /etc/my.cnf 配置文件，在相应段中加入相应的参数字符集，修改完毕后，重启 MySQL 服务即可，具体内容如下：

- [client]字段里加入：default-character-set=utf8。
- [mysqld]字段里加入：character-set-server=utf8。
- [mysql]字段里加入：default-character-set=utf8。

(2) MySQL 命令行中运行如下指令，详情如图 11-4 所示。

```
show variables like '%char%';
SET character_set_client = utf8;
SET character_set_results = utf8;
SET character_set_connection = utf8;
```

```
mysql> SET character_set_client = utf8;
Query OK, 0 rows affected (0.00 sec)

mysql> SET character_set_results = utf8;
Query OK, 0 rows affected (0.00 sec)

mysql> SET character_set_connection = utf8;
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like '%char%';
+--------------------------+----------------------------------+
| Variable_name            | Value                            |
+--------------------------+----------------------------------+
| character_set_client     | utf8                             |
| character_set_connection | utf8                             |
| character_set_database   | utf8                             |
| character_set_filesystem | binary                           |
| character_set_results    | utf8                             |
| character_set_server     | utf8                             |
| character_set_system     | utf8                             |
| character_sets_dir       | /usr/local/mysql55/share/charsets/ |
+--------------------------+----------------------------------+
8 rows in set (0.01 sec)
```

图 11-4　设置 MySQL 数据库字符集

11.5　MySQL 数据库密码管理

MySQL 数据库在使用过程中为了加强安全防范，需要设置密码访问，如何设置密码及密码忘记如何破解呢？如下为设置密码授权、密码修改及密码破解的方法：

（1）MySQL 创建用户及授权。

命令如下：

```
grant all on jfedu.* to test@localhost identified by 'pas';
grant select,insert,update,delete on *.* to test@"%" identified by 'pas';
grant all on jfedu.* to test@'192.168.111.118' identified by 'pas';
```

以上三条 grant 语句授权参数详解如下：

- 授权 localhost 主机通过 test 用户和 pas 密码访问本地的 jfedu 库的所有权限；
- 授权所有主机通过 test 用户和 pas 密码访问本地的 jfedu 库的查询、插入、更新、删除权限；
- 授权 192.168.111.118 主机通过 test 用户和 pas 密码访问本地的 jfedu 库的所有权限。

（2）MySQL 密码破解方法。

在使用 MySQL 数据库中，偶尔会出现密码忘记或者被其他人员修改掉数据库权限的情况，如果需要紧急修改密码，如何破解 MySQL 密码呢？首先停止 MySQL 数据库服务，以跳过权限方式启动，命令如下：

```
/etc/init.d/mysqld stop
/usr/bin/mysqld_safe --user=mysql --skip-grant-tables &
```

MySQL 跳过权限方式启动后，在 shell 终端执行 MySQL 命令并按 Enter 键，进入 MySQL 命令行，如图 11-5 所示。

```
[root@localhost ~]# /usr/bin/mysqld_safe --user=mysql --skip-grant-tables &
[2] 2103
[root@localhost ~]# 141126 08:40:58 mysqld_safe Logging to '/var/log/mysqld.log'.
141126 08:40:58 mysqld_safe Starting mysqld daemon with databases from /var/lib/mysql

[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

图 11-5　跳过权限启动并登录 MySQL

由于 MySQL 用户及密码认证信息存放在 MySQL 库中的 user 表，需进入 MySQL 库，更新相应的密码字段即可，例如将 MySQL 中 root 用户的密码均改为 123456，命令如下，详情如图 11-6 所示。

```
use mysql
update user set password = password('123456') where user = 'root';
```

```
mysql> use mysql
Database changed
mysql> update user set password=password('123456') where user='root';
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

图 11-6　MySQL 密码破解方法

MySQL root 密码修改完，需停止 MySQL 跳过权限表的启动进程，再以正常方式启动 MySQL，再次以新的密码登录即可进入 MySQL 数据库，如图 11-7 所示。

```
[root@localhost ~]# /etc/init.d/mysqld start
Starting MySQL.. SUCCESS!
[root@localhost ~]#
[root@localhost ~]# mysql -uroot -p123456
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.5.20 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

图 11-7　MySQL 正常方式启动

11.6　MySQL 数据库配置文件详解

理解 MySQL 配置文件，可以更快地学习和掌握 MySQL 数据库服务器，以下为 MySQL 配置文件常用参数详解：

- [mysqld]：服务器端配置。
- datadir=/data/mysql：数据目录。
- socket=/var/lib/mysql/mysql.sock：socket 通信设置。
- user=mysql：使用 MySQL 用户启动。
- symbolic-links=0：是否支持快捷方式。
- log-bin=mysql-bin：开启 bin-log 日志。
- server-id = 1：MySQL 服务的 ID。
- auto_increment_offset=1：自增长字段从固定数开始。
- auto_increment_increment=2：自增长字段每次递增的量。
- socket = /tmp/mysql.sock：MySQL 客户程序与服务器之间的本地通信套接字文件。
- port = 3306：指定 MySQL 监听的端口。
- key_buffer = 384MB：key_buffer 是用于索引块的缓冲区大小。
- table_cache = 512：为所有线程打开表的数量。
- sort_buffer_size = 2MB：为每个需要进行排序的线程分配该大小的一个缓冲区。
- read_buffer_size = 2MB：读查询操作所能使用的缓冲区大小。
- query_cache_size = 32MB：指定 MySQL 查询结果缓冲区的大小。
- read_rnd_buffer_size = 8MB：改参数在使用行指针排序之后，随机读。
- myisam_sort_buffer_size = 64MB：MyISAM 表发生变化时重新排序所需的缓冲。
- thread_concurrency = 8：最大并发线程数，取值为服务器逻辑 CPU 数量×2。
- thread_cache = 8：缓存可重用的线程数。
- skip-locking：避免 MySQL 的外部锁定，减少出错几率增强稳定性。
- default-storage-engine=INNODB：设置 MySQL 默认引擎为 InnoDB。
- #mysqld_safe config：MySQL 服务安全配置。
- [mysqld_safe]：MySQL 服务安全启动配置。
- log-error=/var/log/mysqld.log：MySQL 错误日志路径。
- pid-file=/var/run/mysqld/mysqld.pid：MySQL PID 进程文件。
- key_buffer_size = 2048MB：MyISAM 表索引引缓冲区的大小。
- max_connections = 3000：MySQL 最大连接数。
- innodb_buffer_pool_size= 2048MB：InnoDB 内存缓冲数据和索引大小。
- basedir = /usr/local/mysql55/：数据库安装路径。

- [mysqldump]：数据库导出段配置。
- max_allowed_packet =16MB：服务器和客户端发送的最大数据包。

11.7 MySQL 数据库索引案例

MySQL 索引可以用来快速地寻找某些具有特定值的记录，所有 MySQL 索引都以 B-树的形式保存。如果 MySQL 没有索引，执行 select 时会从第一个记录开始扫描整个表的所有记录，直至找到符合要求的记录。如果表中数据有上亿条数据，查询一条数据花费的时间会非常长，索引类似于电子书的目录与页码的对应关系，可加快数据的查找。

如果在需搜索条件的列上创建了索引，MySQL 无须扫描全表记录即可快速得到相应的记录行。如果该表有 1000000 条记录，通过索引查找记录要比全表顺序扫描至少快 100 倍，这就是索引在企业环境中带来的执行速度上的提升。

MySQL 数据库常见索引类型包括：普通索引(normal)、唯一索引(unique)、全文索引(full text)、主键索引(primary key)、组合索引等，以下为每个索引的应用场景及区别：

- 普通索引：normal，使用最广泛。
- 唯一索引：unique，不允许重复的索引，允许有空值。
- 全文索引：full text，只能用于 MyISAM 表，full text 主要用于大量的内容检索。
- 主键索引：primary key 又称为特殊的唯一索引，不允许有空值。
- 组合索引：为提高 MySQL 效率可建立组合索引。

MySQL 数据库表创建各个索引命令，以 t1 表为案例，操作如下：

- 主键索引：ALTER TABLE t1 ADD PRIMARY KEY('column')。
- 唯一索引：ALTER TABLE t1 ADD UNIQUE('column')。
- 普通索引：ALTER TABLE t1 ADD INDEX index_name('column')。
- 全文索引：ALTER TABLE t1 ADD FULLTEXT('column')。
- 组合索引：ALTER TABLE t1 ADD INDEX index_name('column1', 'column2', 'column3')。

t1 表的 id 字段创建主键索引，查看索引是否被创建，然后插入相同的 id，提示报错，如图 11-8 所示。

MySQL 数据库表删除各个索引命令，以 t1 表为案例，操作如下：

```
DROP INDEX index_name ON t1;
ALTER TABLE t1 DROP INDEX index_name;
ALTER TABLE t1 DROP PRIMARY KEY;
```

MySQL 数据库查看表索引，操作如下：

```
show index from t1;
show keys from t1;
```

```
mysql> ALTER TABLE t1 ADD PRIMARY KEY ( `id` )
    -> ;
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql>
mysql> show index from t1;
+-------+------------+----------+--------------+-------------+-----------+--
-+--------+------+------------+---------+---------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | C
 | Packed | Null | Index_type | Comment | Index_comment |
+-------+------------+----------+--------------+-------------+-----------+--
-+--------+------+------------+---------+---------------+
| t1    |          0 | PRIMARY  |            1 | id          | A         |
 | NULL   |      | BTREE      |         |               |
+-------+------------+----------+--------------+-------------+-----------+--
-+--------+------+------------+---------+---------------+
1 row in set (0.00 sec)

mysql>
mysql> insert into t1 values ("3","jfedu");
ERROR 1062 (23000): Duplicate entry '3' for key 'PRIMARY'
```

图 11-8　MySQL 主键索引案例演示

MySQL 数据库索引的缺点如下：

- MySQL 数据库索引虽然能够提高数据库查询速度，但同时会降低更新、删除、插入表的速度，例如对表进行 insert、update、delete 时，update 表 MySQL 不仅要保存数据，还需保存更新索引；
- 建立索引会占用磁盘空间，大表上创建了多种组合索引，索引文件就会占用大量的空间。

11.8　MySQL 数据库慢查询

MySQL 数据库慢查询主要用于跟踪异常的 SQL 语句，可以分析出当前程序里哪些 SQL 语句比较耗费资源，慢查询日志则用来记录在 MySQL 中响应时间超过阀值的语句，具体指运行时间超过 long_query_time 值的 SQL 语句，会被记录到慢查询日志中。

MySQL 数据库默认没有开启慢查询日志功能，需手动在配置文件或者 MySQL 命令行中开启，慢查询日志默认写入磁盘中的文件，也可以将慢查询日志写入到数据库表中。

查看数据库是否开启慢查询，命令如下，详情如图 11-9 所示。

```
show variables like "%slow%";
show variables like "%long_query%";
```

MySQL 慢查询参数详解如下：

- log_slow_queries：关闭慢查询日志功能。
- long_query_time：慢查询超时时间，默认为 10s，MySQL 5.5 以上可以设置微秒。
- slow_query_log：关闭慢查询日志。
- slow_query_log_file：慢查询日志文件。
- slow_launch_time：thread create 时间，单位为秒，如果 thread create 的时间超过了



图 11-9　MySQL 数据库慢查询功能

这个值，该变量 slow_launch_time 的值会加 1。

- log-queries-not-using-indexes：记录未添加索引的 SQL 语句。

开启 MySQL 慢查询日志方法有以下两种：

（1）MySQL 数据库命令行执行命令如下：

```
set global slow_query_log = on;
show variables like "%slow%";
```

（2）编辑 my.cnf 配置文件中添加代码如下：

```
log-slow-queries = /data/mysql/localhost.log
long_query_time = 0.01
log-queries-not-using-indexes
```

慢查询功能开启之后，数据库会自动将执行时间超过设定时间的 SQL 语句添加至慢查询日志文件中，可以通过慢查询日志文件定位执行慢的 SQL，从而对其优化，可以通过 mysqldumpslow 命令行工具分析日志。

执行命令 mysqldumpslow -h 可以查看命令帮助信息，主要参数包括-s 和-t，其中-s 是排序参数，可选项详解如下：

- l：查询锁的总时间。
- r：返回记录数。
- t：查询总时间排序。
- al：平均锁定时间。

- ar：平均返回记录数。
- at：平均查询时间。
- c：计数。
- -t n：显示头 n 条记录。

MySQL 慢查询 mysqldumpslow 按照返回的行数从大到小，查看前 2 行，命令如下，详情如图 11-10 所示。

```
mysqldumpslow －s r －t 2 localhost.log
```

```
[root@localhost mysql]#
[root@localhost mysql]# mysqldumpslow -s r -t 2 localhost.log

Reading mysql slow query log from localhost.log
Count: 4  Time=0.30s (1s)  Lock=0.00s (0s)  Rows=1.2 (5), root[root]@loc
  select * from t1 where id=N

Count: 1  Time=0.23s (0s)  Lock=0.00s (0s)  Rows=1.0 (1), root[root]@loc
  select count(*) from t1

[root@localhost mysql]#
```

图 11-10　mysqldumpslow 以返回记录排序

MySQL 慢查询 mysqldumpslow 按照查询总时间从大到小，查看前 5 行，同时过滤 select 的 SQL 语句，命令如下，详情如图 11-11 所示。

```
mysqldumpslow －s t －t 5 －g "select" localhost.log
```

```
[root@localhost mysql]#
[root@localhost mysql]# mysqldumpslow -s t -t 5 -g "select" localhost.log

Reading mysql slow query log from localhost.log
Count: 4  Time=0.30s (1s)  Lock=0.00s (0s)  Rows=1.2 (5), root[root]@localhost
  select * from t1 where id=N

Count: 1  Time=0.23s (0s)  Lock=0.00s (0s)  Rows=1.0 (1), root[root]@localhost
  select count(*) from t1

Died at /usr/bin/mysqldumpslow line 162, <> chunk 11.
[root@localhost mysql]#
```

图 11-11　mysqldumpslow 以查询总时间排序

11.9　MySQL 数据库优化

MySQL 数据库优化是一项非常重要的工作，而且是一项长期的工作，MySQL 优化三分靠配置文件及硬件资源的优化，七分靠 SQL 语句的优化。

MySQL 数据库具体优化包括：配置文件的优化、SQL 语句的优化、表结构的优化、索引的优化，而配置的优化包括：系统内核、硬件资源、内存、CPU、MySQL 本身配置文件的优化。

硬件上的优化有两种方式：一种是增加内存和提高磁盘读写速度，进而提高 MySQL

数据库的查询、更新的速度；另一种提高 MySQL 性能的方式是使用多块磁盘来存储数据，可以从多块磁盘上并行读取数据，进而提高读取数据的速度。

MySQL 参数的优化，内存中会为 MySQL 保留部分的缓冲区，这些缓冲区可以提高 MySQL 的速度，缓冲区的大小可以在 MySQL 的配置文件中进行设置。

以下为企业级 MySQL 百万量级真实环境配置文件 my.cnf 的内容，用户可以根据实际情况修改，代码如下：

```
[client]
port = 3306
socket = /tmp/mysql.sock
[mysqld]
user = mysql
server_id = 10
port = 3306
socket = /tmp/mysql.sock
datadir = /data/mysql/
old_passwords = 1
lower_case_table_names = 1
character-set-server = utf8
default-storage-engine = MYISAM
log-bin = bin.log
log-error = error.log
pid-file = mysql.pid
long_query_time = 2
slow_query_log
slow_query_log_file = slow.log
binlog_cache_size = 4MB
binlog_format = mixed
max_binlog_cache_size = 16MB
max_binlog_size = 1GB
expire_logs_days = 30
ft_min_word_len = 4
back_log = 512
max_allowed_packet = 64MB
max_connections = 4096
max_connect_errors = 100
join_buffer_size = 2MB
read_buffer_size = 2MB
read_rnd_buffer_size = 2MB
sort_buffer_size = 2MB
query_cache_size = 64MB
table_open_cache = 10000
thread_cache_size = 256
max_heap_table_size = 64MB
tmp_table_size = 64MB
```

```
thread_stack = 192KB
thread_concurrency = 24
local-infile = 0
skip-show-database
skip-name-resolve
skip-external-locking
connect_timeout = 600
interactive_timeout = 600
wait_timeout = 600
# *** MyISAM
key_buffer_size = 512MB
bulk_insert_buffer_size = 64MB
myisam_sort_buffer_size = 64MB
myisam_max_sort_file_size = 1GB
myisam_repair_threads = 1
concurrent_insert = 2
myisam_recover
# *** INNODB
innodb_buffer_pool_size = 64GB
innodb_additional_mem_pool_size = 32MB
innodb_data_file_path = ibdata1:1G; ibdata2:1G:autoextend
innodb_read_io_threads = 8
innodb_write_io_threads = 8
innodb_file_per_table = 1
innodb_flush_log_at_trx_commit = 2
innodb_lock_wait_timeout = 120
innodb_log_buffer_size = 8MB
innodb_log_file_size = 256MB
innodb_log_files_in_group = 3
innodb_max_dirty_pages_pct = 90
innodb_thread_concurrency = 16
innodb_open_files = 10000
#innodb_force_recovery = 4
# *** Replication Slave
read-only
#skip-slave-start
relay-log = relay.log
log-slave-updates
```

11.10　MySQL 数据库集群实战

随着访问量的不断增加，单台 MySQL 数据库服务器压力不断地增加，需要对 MySQL 进行优化和架构改造，如果 MyQSL 优化不能明显改善压力，可以使用高可用、主从复制、读

写分离来、拆分库、拆分表等方法来进行优化。

MySQL 主从复制集群在中小企业、大型企业中被广泛应用，MySQL 主从复制的目的是实现数据库冗余备份，将 master 数据库数据定时同步至 slave 库中，一旦 master 数据库宕机，可以将 Web 应用数据库配置快速切换至 slave 数据库，确保 Web 应用有较高的可用率，MySQL 主从复制架构图如图 11-12 所示。

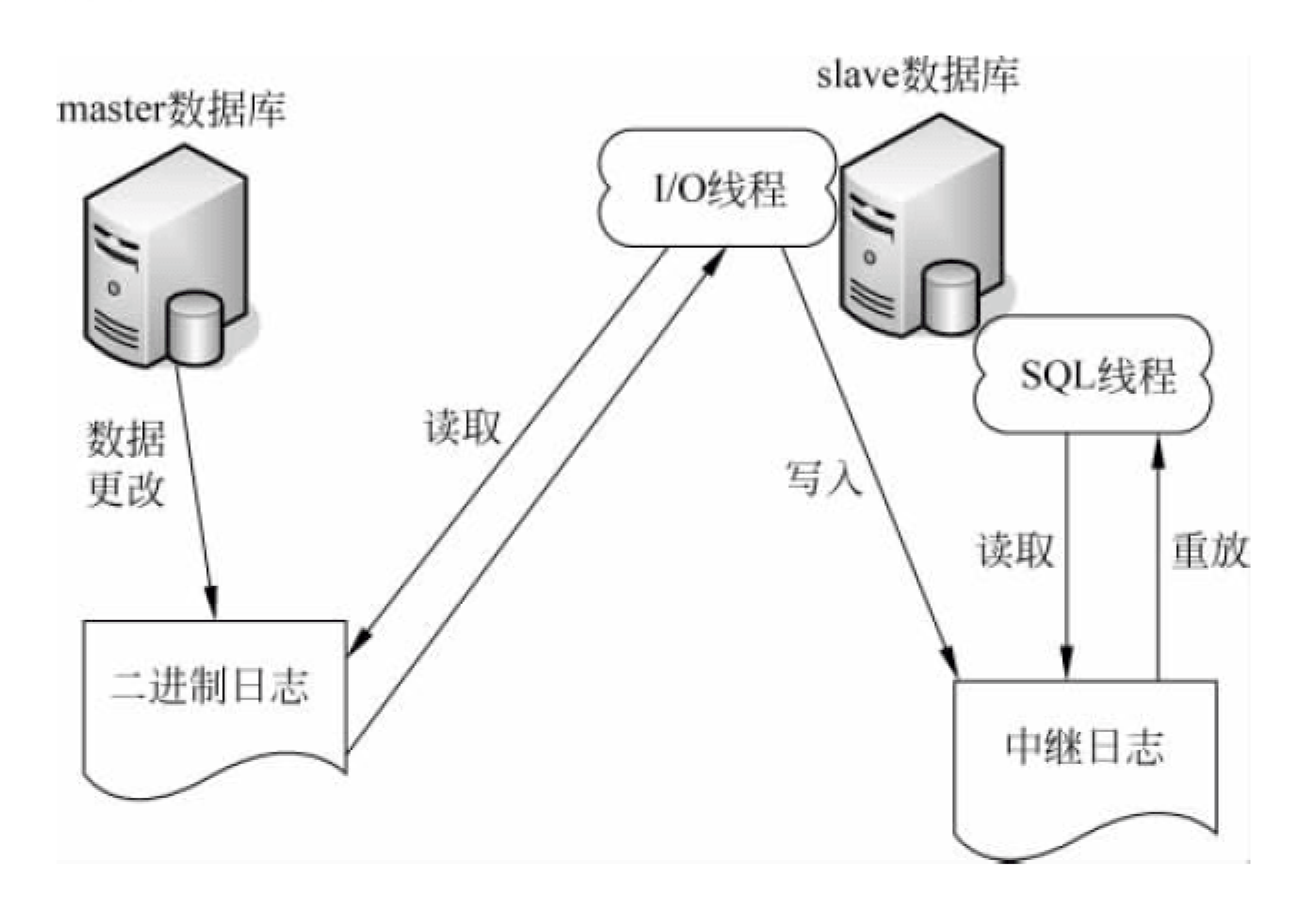


图 11-12　MySQL 主从原理架构图

MySQL 主从复制集群至少需要 2 台数据库服务器，其中一台为 master 库，另外一台为 slave 库，MySQL 主从数据同步是一个异步复制的过程，要实现复制首先需要在 master 上开启 bin-log 日志功能，bin-log 日志用于记录在 master 库中执行的增、删、修改、更新操作的 SQL 语句，整个过程需要开启 3 个线程，分别是 master 开启 I/O 线程，slave 开启 I/O 线程和 SQL 线程，具体主从同步原理详解如下：

- slave 上执行 slave start，slave I/O 线程会通过在 master 创建的授权用户连接上至 master，并请求 master 从指定的文件和位置之后发送 bin-log 日志内容；
- master 接收到来自 slave I/O 线程的请求后，master I/O 线程根据 slave 发送的指定 bin-log 日志 position 点之后的内容，然后返回给 slave 的 I/O 线程；
- 返回的信息中除了 bin-log 日志内容外，还有 master 最新的 bin-log 文件名以及在 bin-log 中的下一个指定更新 position 点；
- slave I/O 线程接收到信息后，将接收到的日志内容依次添加到 slave 端的 relay-log 文件的最末端，并将读取到的 master 端的 bin-log 的文件名和 position 点记录到 master. info 文件中，以便在下一次读取的时候能告知 master 从相应的 bin-log 文件名及最后一个 position 点开始发起请求；
- slave SQL 线程检测到 relay-log 中内容有更新，会立刻解析 relay-log 日志中的内容，将解析后的 SQL 语句在 slave 里执行，执行成功后 slave 库与 master 库数据保持一致。

11.11 MySQL 主从复制实战

MySQL 主从复制环境构建至少需 2 台服务器，可以配置 1 主多从、多主多从，以 1 主 1 从为例，MySQL 主从复制架构实战步骤如下：

（1）系统环境准备。

```
master: 192.168.111.128
slave: 192.168.111.129
```

（2）master 安装及配置。

master 端使用源码安装 MySQL5.5 版本软件后，在/etc/my.cnf 配置文件[mysqld]段中加入如下代码，然后重启 MySQL 服务即可。如果在安装时 cp my-large.cnf /etc/my.cnf，则无须添加如下代码：

```
server-id = 1
log-bin = mysql-bin
```

master 端/etc/my.cnf 完整配置代码如下：

```
[client]
port     = 3306
socket   = /tmp/mysql.sock
[mysqld]
port     = 3306
socket   = /tmp/mysql.sock
skip-external-locking
key_buffer_size = 256MB
max_allowed_packet = 1MB
table_open_cache = 256
sort_buffer_size = 1MB
read_buffer_size = 1MB
read_rnd_buffer_size = 4MB
myisam_sort_buffer_size = 64MB
thread_cache_size = 8
query_cache_size= 16MB
thread_concurrency = 8
log-bin=mysql-bin
binlog_format=mixed
server-id= 1
[mysqldump]
quick
max_allowed_packet = 16MB
[mysql]
no-auto-rehash
```

```
[myisamchk]
key_buffer_size = 128MB
sort_buffer_size = 128MB
read_buffer = 2MB
write_buffer = 2MB
[mysqlhotcopy]
interactive-timeout
```

在 master 数据库服务器命令行中创建 tongbu 用户及密码并设置权限，执行如下命令，查看 bin-log 文件及 position 点，详情如图 11-13 所示。

```
grant replication slave on *.* to 'tongbu'@'%' identified by '123456';
show master status;
```

```
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> grant  replication  slave  on *.* to  'tongbu'@'%'  identified by  '123456';
Query OK, 0 rows affected (0.01 sec)

mysql> show master status;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000028 |      257 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

图 11-13　MySQL master 授权用户

(3) slave 安装及配置。

slave 端使用源码安装 MySQL 5.5 版本软件后，在/etc/my.cnf 配置文件[mysqld]段中加入如下代码，然后重启 MySQL 服务即可。如果在安装时 cp my-large.cnf /etc/my.cnf，则需修改 server-id，master 与 slave 端 server-id 不能一样，slave 端无须开启 bin-log 功能，代码如下：

```
server-id = 2
```

slave 端/etc/my.cnf 完整配置代码如下：

```
[client]
port      = 3306
socket    = /tmp/mysql.sock
[mysqld]
port      = 3306
socket    = /tmp/mysql.sock
skip-external-locking
key_buffer_size = 256MB
max_allowed_packet = 1MB
```

```
table_open_cache = 256
sort_buffer_size = 1MB
read_buffer_size = 1MB
read_rnd_buffer_size = 4MB
myisam_sort_buffer_size = 64MB
thread_cache_size = 8
query_cache_size= 16MB
thread_concurrency = 8
server-id= 2
[mysqldump]
quick
max_allowed_packet = 16MB
[mysql]
no-auto-rehash
[myisamchk]
key_buffer_size = 128MB
sort_buffer_size = 128MB
read_buffer = 2MB
write_buffer = 2MB
[mysqlhotcopy]
interactive-timeout
```

slave 指定 master IP、用户名、密码、bin-log 文件名(mysql-bin.000028)及 position(257),代码如下:

```
change master to
master_host='192.168.111.128',master_user='tongbu',master_password='123456',master_log_
file='mysql-bin.000028',master_log_pos=257;
```

在 slave 中启动 slave start,并执行 show slave status\G 查看 MySQL 主从状态,代码如下:

```
slave start;
show slave status\G
```

查看 slave 端 I/O 线程、SQL 线程状态均为 Yes,代表 slave 已正常连接 master 实现同步,代码如下:

```
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

执行 Show slave status\G,常见参数含义解析如下:

- Slave_IO_State:I/O 线程连接 master 状态。
- Master_User:用于连接 master 的用户。
- Master_Port:master 端监听端口。

- Connect_Retry：主从连接失败，重试时间间隔。
- Master_Log_File：I/O 线程读取的 master 二进制日志文件的名称。
- Read_Master_Log_Pos：I/O 线程已读取的 master 二进制日志文件的位置。
- Relay_Log_File：SQL 线程读取和执行的中继日志文件的名称。
- Relay_Log_Pos：SQL 线程已读取和执行的中继日志文件的位置。
- Relay_Master_Log_File：SQL 线程执行的 master 二进制日志文件的名称。
- Slave_IO_Running：I/O 线程是否被启动并成功地连接到主服务器上。
- Slave_SQL_Running：SQL 线程是否被启动。
- Replicate_Do_DB：指定的同步的数据库列表。
- Skip_Counter：SQL_SLAVE_SKIP_COUNTER 设置的值。
- Seconds_Behind_Master：slave 端 SQL 线程和 I/O 线程之间的时间差距，单位为秒，常被用于主从延迟检查方法之一。

在 master 端创建 mysql_db_test 数据库和 t0 表，命令如下，详情如图 11-14 所示。

```
create database mysql_ab_test charset = utf8;
show databases;
use mysql_ab_test;
create table t0 (id varchar(20),name varchar(20));
show tables;
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database mysql_ab_test charset=utf8;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| mysql_ab_test      |
| test               |
+--------------------+
4 rows in set (0.00 sec)

mysql> use mysql_ab_test;
Database changed
mysql>
mysql> create table t0 (id varchar(20),name varchar(30));
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
+-------------------------+
| Tables_in_mysql_ab_test |
+-------------------------+
| t0                      |
+-------------------------+
1 row in set (0.00 sec)
```

图 11-14　MySQL master 创建数据库和表

Slave 服务器查看是否有 mysql_ab_test 数据库和 t0 的表，如果存在则代表 slave 从 master 复制数据成功，MySQL 主从架构至此配置成功，如图 11-15 所示。

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| mysql_ab_test      |
| test               |
+--------------------+
4 rows in set (0.00 sec)

mysql> use mysql_ab_test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-------------------------+
| Tables_in_mysql_ab_test |
+-------------------------+
| t0                      |
+-------------------------+
1 row in set (0.00 sec)
```

图 11-15　MySQL slave 自动同步数据

在 master 服务器的 t0 表中插入两条数据，命令如下，在 slave 查看是否已同步，master 上执行详情如图 11-16 所示。

```
insert into t0 values ("001","wugk1");
insert into t0 values ("002","wugk2");
select * from t0;
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use mysql_ab_test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> insert into t0 values ("001","wugk1");
Query OK, 1 row affected (0.00 sec)

mysql> insert into t0 values ("002","wugk2");
Query OK, 1 row affected (0.00 sec)

mysql> select * from t0;
+------+-------+
| id   | name  |
+------+-------+
| 001  | wugk1 |
| 002  | wugk2 |
+------+-------+
2 rows in set (0.00 sec)

mysql>
```

图 11-16　MySQL master insert 数据

Slave 端执行查询命令，如图 11-17 所示，表示 master 端插入的 SQL 数据已经同步到 slave 端。

```
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use mysql_ab_test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from t0;
+------+-------+
| id   | name  |
+------+-------+
| 001  | wugk1 |
| 002  | wugk2 |
+------+-------+
2 rows in set (0.00 sec)
```

图 11-17　MySQL slave 数据已同步

11.12　MySQL 主从同步排错思路

MySQL 主从同步集群在生产环境使用时，如果主从服务器之间网络通信条件差或者数据库数据量非常大，容易导致 MySQL 主从同步延迟。

MySQL 主从产生延迟之后，一旦主库宕机，会导致部分数据没有及时同步至从库，重新启动主库，会导致从库与主库同步错误，快速恢复主从同步关系有如下两种方法：

（1）忽略错误后，继续同步。

此种方法适用于主从库数据内容相差不大的情况。

master 端执行如下命令，将数据库设置为全局读锁，不允许写入新数据。

```
flush tables with read lock;
```

slave 端停止 slave I/O 及 SQL 线程，同时将同步错误的 SQL 跳过 1 次，跳过错误会导致数据不一致，启动 start slave，同步状态恢复，命令如下：

```
stop slave;
set global sql_slave_skip_counter = 1;
start slave;
```

（2）重新做主从同步，使数据完全同步。

此种方法适用于主从库数据内容相差很大的情况。

master 端执行如下命令，将数据库设置全局读锁，不允许写入新数据。

```
flush tables with read lock;
```

master 端基于 mysqldump、xtrabackup 工具对数据库进行完整备份，也可以用 shell 脚本或 python 脚本实现定时备份，备份成功之后，将完整的数据导入至从库，重新配置主从关系，当 slave 端的 I/O 线程、SQL 线程均为 Yes 之后，最后将 master 端读锁解开即可，解锁命令如下：

```
unlock tables;
```

第 12 章



LAMP 企业架构实战

Linux 下 LAMP(Linux+Apache+MySQL/MariaDB+Perl/PHP/Python)是一组用来搭建动态网站的开源软件架构,本身是各自独立的软件服务,放在一起使用,拥有了越来越高的兼容度,共同组成了一个强大的 Web 应用程序平台。

本章向读者介绍互联网主流企业架构 LAMP 应用案例、PHP 解释性语言详解、LAMP 组合通信原理、LAMP 企业源码架设、LAMP 拓展及使用 Redis 提升 LAMP 性能优化等内容。

12.1 LAMP 企业架构简介

随着开源潮流的蓬勃发展,开放源代码的 LAMP 已经与 J2EE 和.Net 商业软件形成三足鼎立之势,并且该软件开发的项目在软件方面的投资成本较低,因此受到整个 IT 界的关注。LAMP 架构受到大多数中小企业的运维、DBA、程序员的青睐,Apache 默认只能发布静态网页,而 LAMP 组合可以发布静态和 PHP 动态页面。

静态页面通常指不与数据库发生交互的页面,是一种基于 W3C 规范的一种网页书写格式,是一种统一协议语言,所以称为静态网页。静态页面被设计好之后,一般很少去修改,不随着浏览器参数改变而内容改变,需注意的是动态的图片也是属于静态文件。从 SEO 角度来讲,HTML 页面更有利于搜索引擎的爬行和收录。常见的静态页面以.html、.gif、.jpg、.jpeg、.bmp、.png、.ico、.txt、.js、.css 等结尾。

动态页面通常指与数据库发生交互的页面,内容展示丰富,功能非常强大,实用性广。从 SEO 角度来讲,搜索引擎很难全面的爬行和收录动态网页,因为动态网页会随着数据库的更新、参数的变更而发生改变,常见的动态页面以.jsp、.php、.do、.asp、.cgi、.apsx 等结尾。

12.2 Apache 与 PHP 工作原理

LAMP 企业主流架构最重要的三个环节:一是 Apache Web 服务器;二是 PHP(hypertext preprocessor);三是 MySQL 数据库。

Apache Web 服务器主要是基于多模块工作，依赖 PHP SAPI 处理方式中的 PHP_MODULE 去解析 PHP 结尾的文件，如图 12-1 所示。

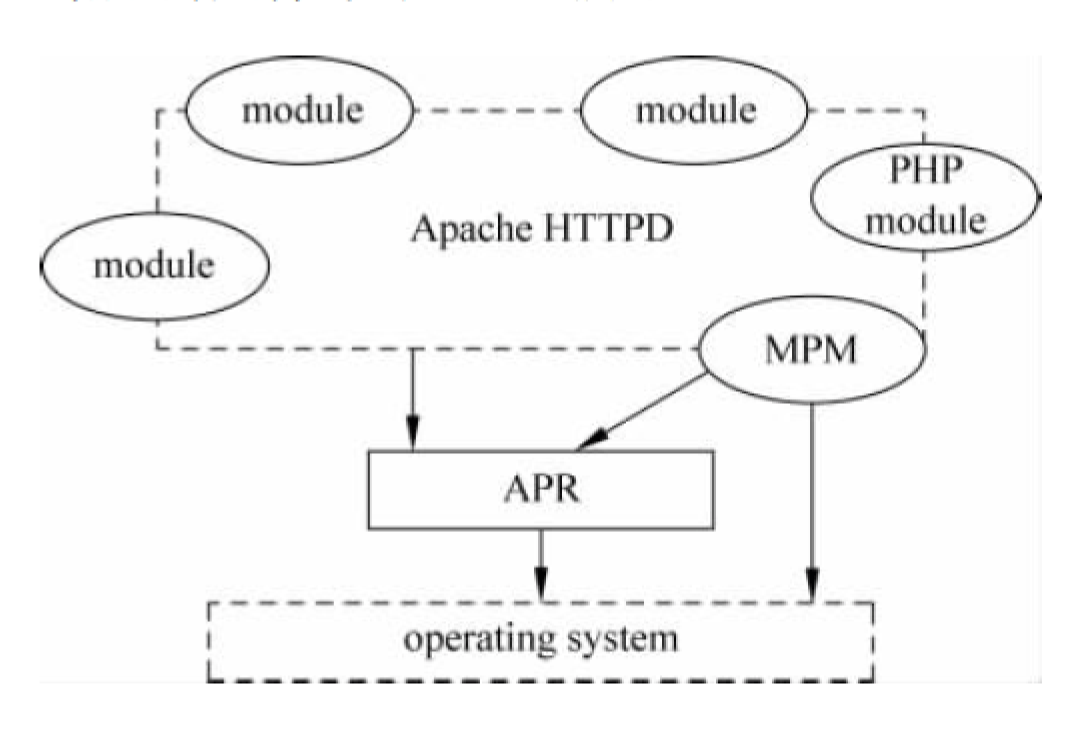


图 12-1 Apache+PHP 多模块工作原理

PHP 是一种适用于 Web 开发的动态语言，PHP 语言内核基于 C 语言实现包含大量组件的软件框架，是一种功能强大的解释型脚本语言。PHP 底层运行机制如图 12-2 所示。

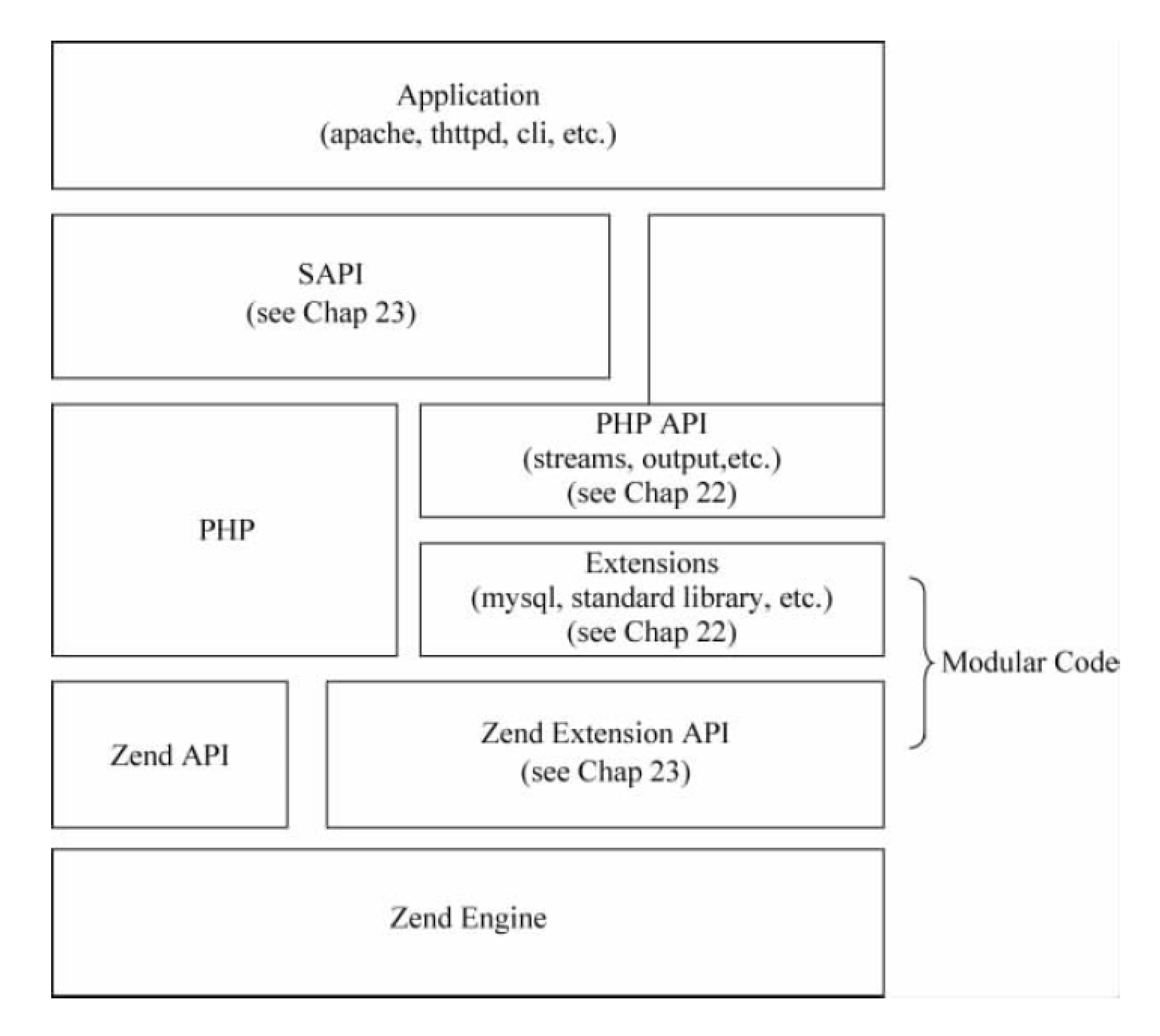


图 12-2 PHP 底层处理机制

PHP 底层工作原理包括以下 4 个部分。

- Zend 引擎：属于 PHP 内核部分，它负责将 PHP 代码解析为可执行 opcode 的处理并实现相应的处理方法、实现基本的数据结构、内存分配及管理、提供了相应的 API

方法供外部调用，是一切的核心，所有的外围功能均围绕 Zend 实现。

- Extensions：围绕着 Zend 引擎，Extensions 通过组件的方式提供各种基础服务，各种内置函数、标准库等都是通过 Extensions 来实现的。
- SAPI：服务端应用编程接口（server application programming interface，SAPI），SAPI 通过一系列钩子函数，基于 SAPI 可以让 PHP 与外部进行数据交互。
- 常见的 SAPI 编程接口处理方法包括以下几种：apache2handler——以 Apache 作为 WebServer，采用 MOD_PHP 模式运行时候的处理方式；cgi——WebServer 和 PHP 直接的另一种交互方式，FastCGI 协议；cli——命令行调用的应用模式。
- APP 代码应用：又称为 PHP 代码程序，基于 SAPI 接口生成不同的应用模式，从而被 PHP 引擎解析。

当用户在浏览器地址中输入域名或者域名和 PHP 页面，向 Web 服务器 Apache 发起 HTTP 请求，Web 服务器接受该请求，并根据其后缀判断如果请求的页面是以.php 结尾，Web 服务器从硬盘或者内存中取出该 PHP 文件，将其发送给 PHP 引擎程序。

PHP 引擎程序将会对 Web 服务器传送过来的文件进行扫描并根据命令从后台读取、处理数据、并动态地生成相应的 HTML 页面。然后 PHP 引擎程序将生成的 HTML 页面返回给 Web 服务器，最终 Web 服务器将 HTML 页面返回给客户端浏览器，浏览器基于 MIME 类型进行解析展示给用户。

12.3 LAMP 企业安装配置

构建 LAMP 架构有两种方法：一是使用 YUM 在线安装；另外一种是基于 LAMP 源码编译安装，YUM 在线安装方法如下：

```
yum install httpd httpd-devel mysql mysql-server mysql-devel php php-devel php-mysql -y
service httpd restart
service mysqld restart
```

YUM 方式安装简单、快捷，如果需要添加扩展的功能和模块，需使用源码方式来编译安装 LAMP。以下为 LAMP 源码编译安装的步骤：

(1) Apache Web 安装，先安装 apr、apr-utils 库包。

```
yum install apr-devel apr-util-devel -y;
cd /usr/src ;
wget http://mirror.bit.edu.cn/apache/httpd/httpd-2.2.31.tar.gz
tar xzf httpd-2.2.31.tar.gz
cd httpd-2.2.31
./configure --prefix=/usr/local/apache --enable-so --enable-rewrite
make
make install
```

(2) MySQL 数据库安装，基于 MySQL 5.5 编译安装，通过 cmake、make、make install

三个步骤实现。

```
yum install cmake ncurses - devel ncurses  - y
cmake . - DCMAKE_INSTALL_PREFIX = /usr/local/mysql55 \
 - DMYSQL_UNIX_ADDR = /tmp/mysql.sock \
 - DMYSQL_DATADIR = /data/mysql \
 - DSYSCONFDIR = /etc \
 - DMYSQL_USER = mysql \
 - DMYSQL_TCP_PORT = 3306 \
 - DWITH_XTRADB_STORAGE_ENGINE = 1 \
 - DWITH_INNOBASE_STORAGE_ENGINE = 1 \
 - DWITH_PARTITION_STORAGE_ENGINE = 1 \
 - DWITH_BLACKHOLE_STORAGE_ENGINE = 1 \
 - DWITH_MYISAM_STORAGE_ENGINE = 1 \
 - DWITH_READLINE = 1 \
 - DENABLED_LOCAL_INFILE = 1 \
 - DWITH_EXTRA_CHARSETS = 1 \
 - DDEFAULT_CHARSET = utf8 \
 - DDEFAULT_COLLATION = utf8_general_ci \
 - DEXTRA_CHARSETS = all \
 - DWITH_BIG_TABLES = 1 \
 - DWITH_DEBUG = 0
make
make install
```

将源码安装的 MySQL 数据库服务设置为系统服务，可以使用 chkconfig 管理，并启动 MySQL 数据库，代码如下：

```
cd /usr/local/mysql55/
\cp support - files/my - large.cnf /etc/my.cnf
\cp support - files/mysql.server /etc/init.d/mysqld
chkconfig  -- add mysqld
chkconfig  -- level 35 mysqld on
mkdir  - p /data/mysql
useradd mysql
/usr/local/mysql55/scripts/mysql_install_db  -- user = mysql  -- datadir = /data/mysql/
 -- basedir = /usr/local/mysql55/
ln  - s /usr/local/mysql55/bin/ *  /usr/bin/
service mysqld restart
```

(3) PHP 服务安装，PHP 需与 Apache、MySQL 进行整合，参数命令如下，详情如图 12-3 所示。

```
cd /usr/src
wget http://mirrors.sohu.com/php/php - 5.3.28.tar.bz2
tar jxf php - 5.3.28.tar.bz2
cd php - 5.3.28 ;
```

```
./configure --prefix=/usr/local/php5 --with-config-file-path=/usr/local/php5/etc
--with-apxs2=/usr/local/apache/bin/apxs --with-mysql=/usr/local/mysql55/
```

```
Installing PHP CLI man page:      /usr/local/php5/man/man1/
Installing build environment:     /usr/local/php5/lib/php/build/
Installing header files:          /usr/local/php5/include/php/
Installing helper programs:       /usr/local/php5/bin/
  program: phpize
  program: php-config
Installing man pages:             /usr/local/php5/man/man1/
  page: phpize.1
  page: php-config.1
Installing PEAR environment:      /usr/local/php5/lib/php/
[PEAR] Archive_Tar    - installed: 1.3.11
[PEAR] Console_Getopt - installed: 1.3.1
warning: pear/PEAR requires package "pear/Structures_Graph" (recommended version 1.0.4)
warning: pear/PEAR requires package "pear/XML_Util" (recommended version 1.2.1)
[PEAR] PEAR           - installed: 1.9.4
Wrote PEAR system config file at: /usr/local/php5/etc/pear.conf
You may want to add: /usr/local/php5/lib/php to your php.ini include_path
[PEAR] Structures_Graph- installed: 1.0.4
[PEAR] XML_Util       - installed: 1.2.1
/usr/src/php-5.3.28/build/shtool install -c ext/phar/phar.phar /usr/local/php5/bin
ln -s -f /usr/local/php5/bin/phar.phar /usr/local/php5/bin/phar
Installing PDO headers:          /usr/local/php5/include/php/ext/pdo/
[root@node2 php-5.3.28]#
```

图 12-3　LAMP 源码编译整合

（4）Apache+PHP 源码整合。

为了能让 Apache 识别 PHP 文件，需要将 PHP 安装完成后生成的 libphp5.so 模块与 Apache 进行整合，vim httpd.conf 编辑配置文件，加入如下代码：

```
LoadModule       php5_module modules/libphp5.so
AddType          application/x-httpd-php .php
DirectoryIndex   index.php index.html index.htm
```

（5）测试 Apache+PHP 环境。

创建 PHP 测试页面，在/usr/local/apache/htdocs 目录下创建 index.php 测试文件，执行如下命令：

```
cat >/usr/local/apache/htdocs/index.php << EOF
<?php
phpinfo();
?>
EOF
```

重新启动 Apache 服务，浏览器输入 Apache Web 的 IP 访问，如图 12-4 所示，即代表 LAMP 源码环境整合成功。

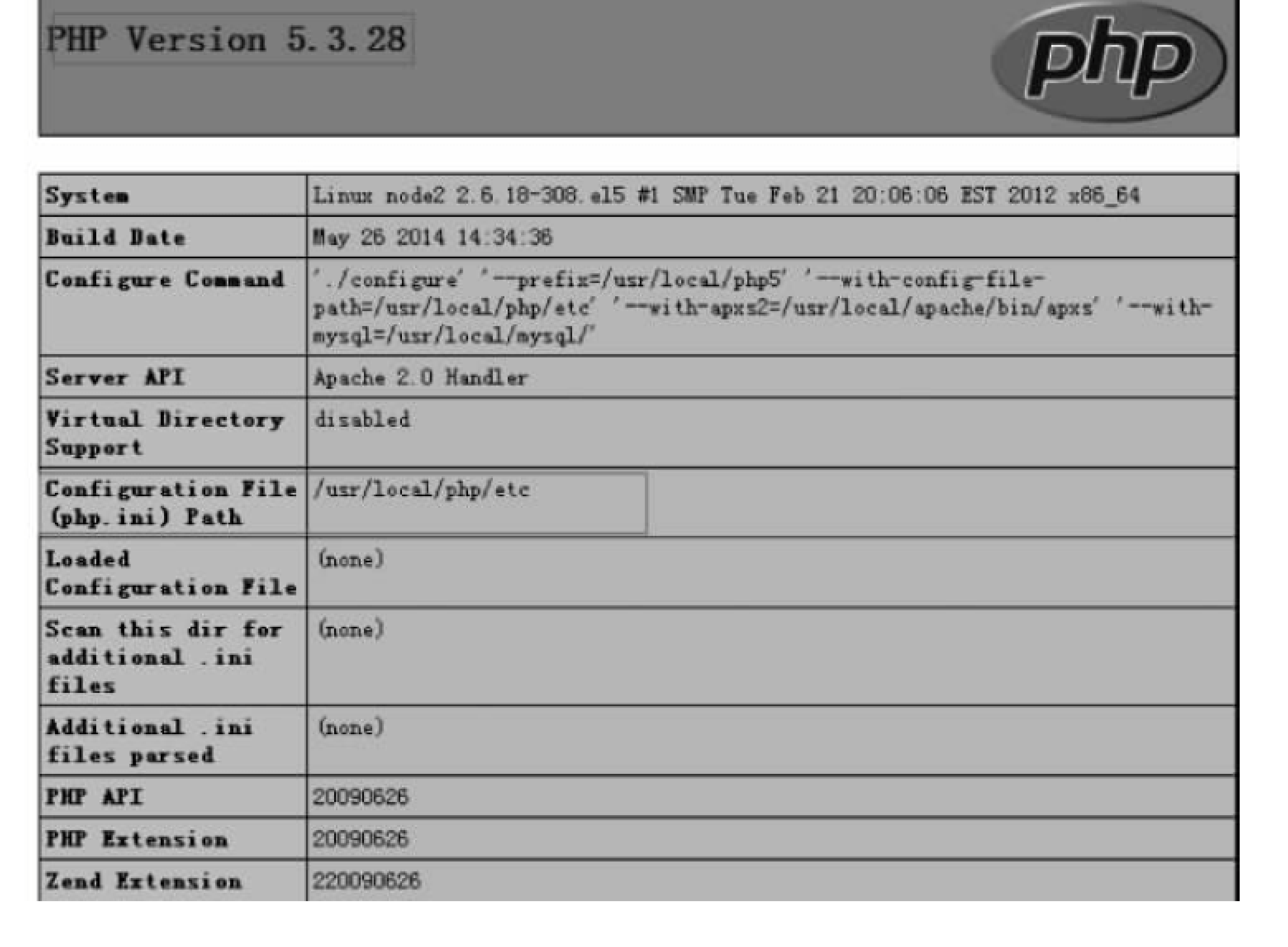
（6）Discuz PHP 论坛安装。

LAMP 源码整合完毕之后，Discuz 官网下载 Discuz 开源 PHP 软件包，将软件包解压至 Apache htdocs 发布目录，命令如下：

```
cd /usr/src ;
wget http://download.comsenz.com/DiscuzX/3.1/Discuz_X3.1_SC_UTF8.zip
unzip Discuz_X3.1_SC_UTF8.zip -d /usr/local/apache/htdocs/
cd /usr/local/apache/htdocs/ ; \mv upload/* .
```

```
chmod 757 -R data/ uc_server/ config/ uc_client/
```

通过浏览器访问 Apache Web IP,如图 12-5 所示,单击"我同意"按钮。



PHP Version 5.3.28

System	Linux node2 2.6.18-308.el5 #1 SMP Tue Feb 21 20:06:06 EST 2012 x86_64
Build Date	May 26 2014 14:34:38
Configure Command	'./configure' '--prefix=/usr/local/php5' '--with-config-file-path=/usr/local/php/etc' '--with-apxs2=/usr/local/apache/bin/apxs' '--with-mysql=/usr/local/mysql/'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/php/etc
Loaded Configuration File	(none)
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626

图 12-4　Apache+PHP 测试页面



图 12-5　Discuz 安装界面(1)

进入如图 12-6 所示界面，数据库安装，如果不存在则需要新建数据库并授权。

Discuz! 安装向导
Discuz!X3.1 简体中文 UTF8 版 20140301
3. 安装数据库
正在执行数据库安装
检查安装环境 设置运行环境 创建数据库 安装
填写数据库信息
数据库服务器: localhost 数据库服务器地址，一般为 localhost
数据库名: discuz
数据库用户名: root
数据库密码: 123456
数据表前缀: pre_ 同一数据库运行多个论坛时，请修改前缀
系统信箱 Email: admin@admin.com 用于发送程序错误报告

图 12-6　Discuz 安装界面(2)

MySQL 数据库命令行中创建 PHP 连接 MySQL 的用户及密码，命令如下：

```
create database discuz charset = utf8;
grant all on discuz. * to root@'localhost' identified by "123456";
```

单击“下一步”按钮，直至安装完成，浏览器自动跳转至如图 12-7 所示界面。



图 12-7　Discuz 安装界面(3)

12.4　LAMP 企业架构拓展实战

LAMP 服务安装至单台服务器，随着用户访问量不断地增加，单台服务器压力逐渐增加，那么如何优化 LAMP 架构，如何拆分 LAMP 架构，怎么把 Apache 和 MySQL 分开放在

不同的机器上呢?

LAMP 架构拆分的目的在于缓解单台服务器的压力,可以将 PHP、MySQL 单独安装至多台服务器,本节将实现 LAP+MySQL 的架构,也即是把 MySQL 单独拆分出去。部署方法有以下两种:

(1) YUM 安装 LAMP 多机方案。

在 Apache Web 服务器只需执行如下代码:

```
yum install httpd httpd-devel php-devel php php-mysql -y
```

在 MySQL 数据库服务器只需执行如下代码:

```
yum install mysql-server mysql mysql-devel mysql-libs -y
```

(2) 源码安装 LAMP 多机方案。

源码安装 LAMP 多机方式,Apache Web 服务与 MySQL 数据库服务分别部署在不同的服务器即可,PHP 与 Apache 服务部署在一台服务器,PHP 编译参数时加入如下代码进行 LAMP 的整合,mysqlnd 为 PHP 远程连接 MySQL 数据库服务器的一种方式。

```
./configure --prefix=/usr/local/php5 \
--with-mysql=mysqlnd --with-mysqli=mysqlnd --with-pdo-mysql=mysqlnd \
--with-apxs2=/usr/local/apache/bin/apxs
make
make install
```

12.5　LAMP+Redis 企业实战

LAMP 在企业生产环境中,除了将 MySQL 单独部署在其他服务器,由于 MySQL 数据库压力会很大,还会使用 MySQL 主从复制及读写分离架构,同时会对 PHP 网站进行调优。PHP 的优化手段包括:PHP 代码本身优化、PHP 配置文件优化、为 PHP 添加缓存模块、将 PHP 网站数据存入缓存等。

12.5.1　Redis 入门简介

Redis 是一个开源的使用 ANSI C 语言编写、支持网络、可基于内存亦可持久化的日志型、key-value 数据库,并提供多种语言的 API。Redis 是一个 key-value 存储系统。

Redis 支持存储的 value 类型包括 string(字符串)、list(链表)、set(集合)、zset(有序集合)和 hash(哈希类型)等。

Redis 是一种高级 key-value 数据库,它跟 Memcached 类似,不过 Redis 的数据可以持久化,而且支持的数据类型更丰富,有字符串、链表、集合和有序集合。同时 Redis 支持在服务器端计算集合的并、交和补集(difference)等,还支持多种排序功能。Redis 也被看成是一个数据结构服务器。

Redis 很大程度补偿了 Memcached key-value 存储的不足，在部分场合可以对关系数据库起到很好的补充作用。Redis 提供了 Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang 等客户端，方便易用，得到 IT 人的青睐。

Redis 支持主从同步，数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得 Redis 可执行单层树复制，由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录，同步对读取操作的可扩展性和数据冗余很有帮助。

使用 Redis 的互联网企业有京东、百度、腾讯、阿里巴巴、新浪、图吧、研修网等，目前的主流数据库功能对比如表 12-1 所示。

表 12-1 常见数据库功能对比

名　称	数据库类型	数据存储选项	操作类型	备　注
Redis	内存存储，NoSQL 数据库	支持字符串、列表、集合、散列表、有序集合	增、删、修改、更新	支持分布式集群、主从同步及高可用、单线程
Memcached	内存缓存数据库，键值对	键值之间的映射	增、删、修改、更新	支持多线程
MySQL	典型关系数据库，RDBMS	数据库由多表主成，每张表包含多行	增、删、修改、更新	支持 ACID 性质
PostgreSQL	典型关系数据库，RDBMS	数据库由多表主成，每张表包含多行	增、删、修改、更新	支持 ACID 性质
MongoDB	硬盘存储，NoSQL 数据库	数据库包含多个表	增、删、修改、更新	主从复制，分片，副本集、空间索引

12.5.2 LAMP＋Redis 工作机制

LAMP＋Redis 工作机制：用户通过浏览器访问 LAMP 网站，并以用户名和密码登录到网站，默认 Redis 缓存中没有该用户名和密码对应列表，PHP 程序会读取 MySQL 数据库中的用户名和密码，然后将用户名和密码缓存至 Redis 中，下次用户通过浏览器再次使用同样的用户名和密码登录网站，PHP 无须从数据库中读取该用户和密码信息，而是直接优先从 Redis 缓存中读取并返回，从而减轻 MySQL 数据库的压力。

Redis 除了可以缓存用户名、密码，还可以缓存 PHP 论坛各种数据，例如用户帖子、用户动态等，如图 12-8 所示。

要实现将 LAMP PHP 网站相关数据存入 Redis，需要一台 Redis 服务器、PHP-Redis 连接驱动、PHP 代码配置等。

12.5.3 LAMP＋Redis 操作案例

LAMP PHP 连接 Redis，首先需安装 Redis 服务，安装连接驱动，然后修改 PHP 网站配

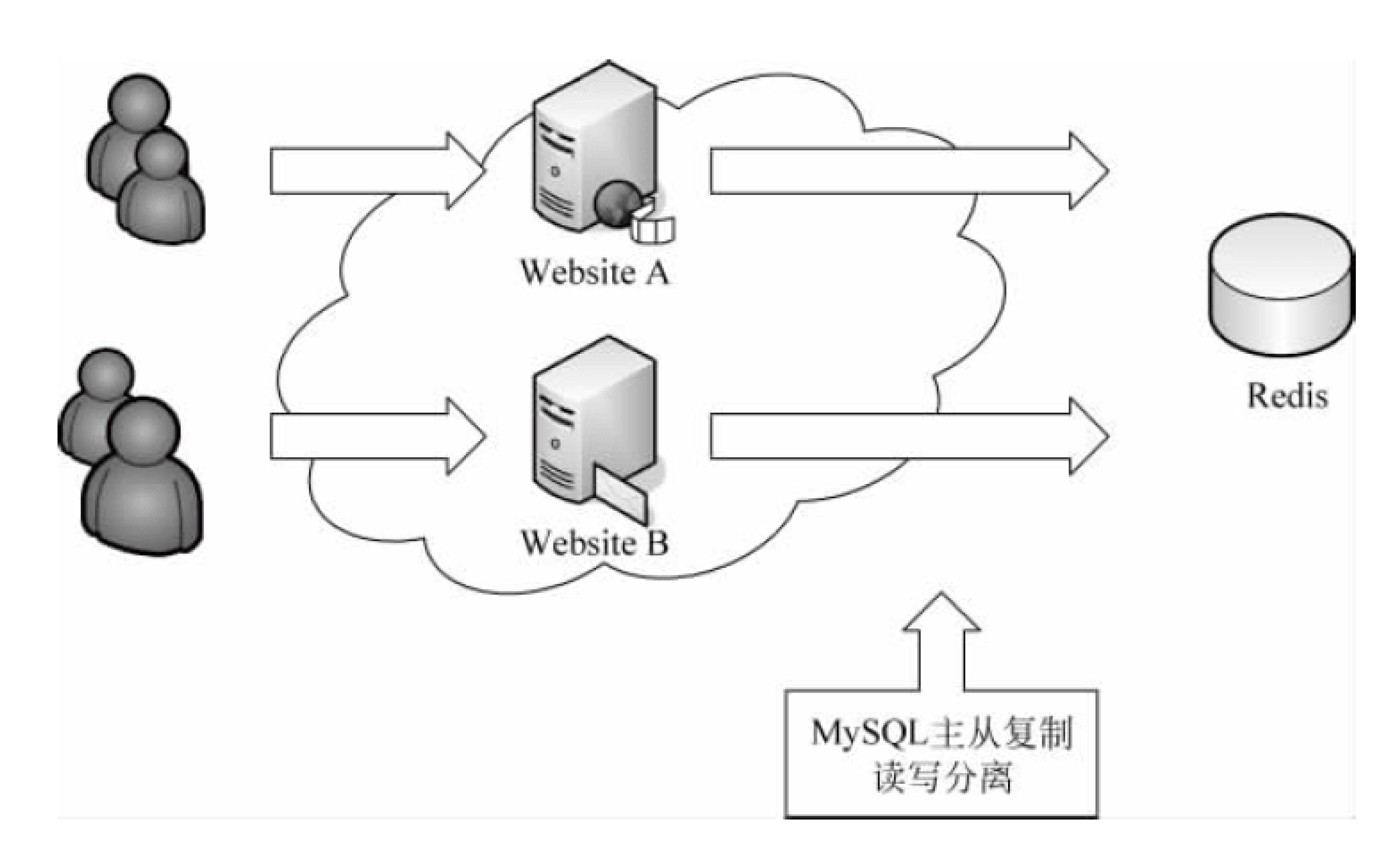


图 12-8　LAMP+Redis 架构流程图

置文件，具体操作步骤如下：

(1) LAMP+Redis 实战环境配置，详细配置如下：

- LAMP 服务器：192.168.149.128。
- Redis 主库：192.168.149.129。
- Redis 从库：192.168.149.130。

(2) 192.168.149.129 服务器安装部署 Redis 服务，代码如下：

```
wget http://download.redis.io/releases/redis-2.8.13.tar.gz
tar zxf redis-2.8.13.tar.gz
cd redis-2.8.13
make PREFIX=/usr/local/redis install
cp redis.conf/usr/local/redis/
```

将/usr/local/redis/bin/目录加入至环境变量配置文件/etc/profile 末尾，然后 shell 终端执行 source /etc/profile 让环境变量生效，代码如下：

```
export PATH=/usr/local/redis/bin:$PATH
```

nohup 后台启动及停止 Redis 服务命令，代码如下：

```
nohup /usr/local/redis/bin/redis-server /usr/local/redis/redis.conf &
/usr/local/redis/bin/redis-cli -p 6379 shutdown
```

(3) 安装 PHP-Redis 连接驱动。

要确保 PHP 能够连接 Redis 缓存服务器，需添加 PHP-Redis 扩展程序，也即是添加 PHP 安装 ext 扩展模块，添加方法如下：

```
wget https://github.com/phpredis/phpredis/archive/3.1.2.tar.gz
tar xzf 3.1.2.tar.gz
```

```
cd phpredis-3.1.2/
./configure --with-php-config=/usr/local/php5/bin/php-config --enable-redis
make
make install
```

修改 vim /usr/local/php/lib/php.ini 配置文件，添加 redis.so 模块，代码如下：

```
extension_dir = "/usr/local/php5/lib/php/extensions/no-debug-zts-20090626"
extension=redis.so
```

重启 Apache 服务，写入 phpinfo 测试页面，通过浏览器访问，如图 12-9 所示，检查到存在 Redis 模块即可。

redis

Redis Support	enabled
Redis Version	3.1.2
Available serializers	php

Reflection

Reflection	enabled
Version	$Revision: 300393 $

图 12-9　PHP-Redis 模块添加

（4）LAMP+Redis 缓存测试。

登录 192.168.149.128 Web 服务器，修改 Discuz PHP 网站发布/usr/local/apache2/htdcos 目录全局配置文件 config_global.php，查找 CONFIG MEMORY 段，将['redis']['server']后改为 Redis 主服务器的 IP，即改为 192.168.149.129，如图 12-10 所示。

```
// -------------------------  CONFIG MEMORY  ----------------
$_config['memory']['prefix'] = 'IOKLan_';
$_config['memory']['redis']['server'] = '192.168.149.129';
$_config['memory']['redis']['port'] = 6379;
$_config['memory']['redis']['pconnect'] = 1;
$_config['memory']['redis']['timeout'] = '0';
$_config['memory']['redis']['requirepass'] = '';
$_config['memory']['redis']['serializer'] = 1;

$_config['memory']['memcache']['server'] = '';
$_config['memory']['memcache']['port'] = 11211;
$_config['memory']['memcache']['pconnect'] = 1;
$_config['memory']['memcache']['timeout'] = 1;
```

图 12-10　PHP-Redis 配置文件修改

通过浏览器访问 Apache PHP 论坛网站，同时登录 Redis 服务器，执行命令 redis-cli 进入 Redis 命令行，运行命令 KEYS *，如图 12-11 所示，存在以 IOKLAN 开头的 key，则证明 Redis 成功缓存 LAMP+Discuz 网站信息数据。

（5）测试 Redis 缓存是否生效。

访问 LAMP+Discuz 网站，创建论坛测试用户 jfedu666，密码 jfedu666，此时用户数据

第一次注册，用户名和密码会写入到 MySQL 数据库表中，同时该数据也会写入到 Redis 缓存，如图 12-12 所示。

```
[root@www-jfedu-net ~]# redis-cli
127.0.0.1:6379>
127.0.0.1:6379> KEYS *
 1) "IOkLan_forum_index_page_1"
 2) "IOkLan_style_default"
 3) "IOkLan_magic"
 4) "IOkLan_usergroups"
 5) "IOkLan_creditrule"
 6) "IOkLan_common_member_field_home_2"
 7) "IOkLan_usergroup_7"
 8) "IOkLan_userstats"
 9) "IOkLan_cronnextrun"
10) "IOkLan_plugin"
11) "IOkLan_common_member_count_1"
12) "IOkLan_onlinelist"
13) "IOkLan_adminmenu"
14) "IOkLan_onlinerecord"
15) "IOkLan_common_member_field_home_1"
```

图 12-11 Redis 缓存 LAMP KEYS 数据

*用户名: jfedu666

*密码: ••••••••

*确认密码: ••••••••

*Email: jfedu666@qq.com

*验证码: eth7 换一个

输入下图中的字符

ETH7

(a) 创建论坛用户和密码

```
mysql>
mysql>
mysql> select uid,email,username,password from pre_common_member;
+-----+---------------------+------------+----------------------------------
| uid | email               | username   | password
+-----+---------------------+------------+----------------------------------
|   1 | admin@admin.com     | admin      | b2698f250316a7b6f6b8b8df80bae
|   2 | jfedu@qq.com        | jfedu      | 15a67a7ada8faf14e164184eb1313
|   3 | jfedujfedu@163.com  | jfedujfedu | c730c9b640b5c63479e34011b979c
|   5 | jfedu666@qq.com     | jfedu666   | a777c022d7c92fbcb3e5b25e3a8398
+-----+---------------------+------------+----------------------------------
4 rows in set (0.00 sec)
```

(b) MySQL数据库用户查询

```
29) "IOKLan_creditrule"
30) "IOKLan_fields_required"
31) "IOKLan_common_member_5"
32) "IOKLan_userapp"
redis 127.0.0.1:6379> get IOKLan_common_member_5
"a:23:{s:3:\"uid\";s:1:\"5\";s:5:\"email\";s:15:\"jfedu666@qq.com
32:\"a777c022d7c92fbcb3e5b25e3a83983a\";s:6:\"status\";s:1:\"0\";
:\"0\";s:16:\"videophotostatus\";s:1:\"0\";s:7:\"adminid\";s:1:\
\"0\";s:11:\"extgroupids\";s:0:\"\";s:7:\"regdate\";s:10:\"149705
1:\"0\";s:10:\"timeoffset\";s:4:\"9999\";s:5:\"newpm\";s:1:\"0\";
2:\"allowadmincp\";s:1:\"0\";s:18:\"onlyacceptfriendpm\";s:1:\"0\
"

redis 127.0.0.1:6379>
```

(c) Redis缓存测试案例

图 12-12 测试 Redis 缓存

将 jfedu666 从 MySQL Discuz 库 pre_common_member 中删除，通过该用户依然可以正常登录 Web 网站，则证明此时数据读取的是 Redis 缓存服务器，如图 12-13 所示。



图 12-13　测试 Redis 缓存是否生效

12.6　Redis 配置文件详解

Redis 是一个内存数据库，redis.conf 常用参数的详解如下：

```
# daemonize no Linux shell 终端运行 Redis,改为 yes 即后台运行 Redis 服务
daemonize yes
```

```
# 当运行多个 Redis 服务时,需要指定不同的 pid 文件和端口
pidfile /var/run/redis_6379.pid
# 指定 Redis 运行的端口,默认是 6379
port 6379
# 在高并发的环境中,为避免慢客户端的连接问题,需要设置一个高速后台日志
tcp-backlog 511
# 指定 Redis 只接收来自于该 IP 地址的请求,如果不进行设置,那么将处理所有请求
# bind 192.168.1.100 10.0.0.1
# bind 127.0.0.1
#设置客户端连接时的超时时间,单位为 s.当客户端在这段时间内没有发出任何指令,那么关闭该连接
timeout 0
# 在 Linux 上,指定值(s)用于发送 ACKs 的时间.注意关闭连接需要双倍的时间.默认为 0
tcp-keepalive 0
# Redis 总共支持 4 个日志级别: debug、verbose、notice、warning,默认为 verbose
# debug: 记录很多信息,用于开发和测试
# varbose: 有用的信息,不像 debug 会记录那么多
# notice: 普通的 verbose,常用于生产环境
# warning: 只有非常重要或者严重的信息会记录到日志
loglevel notice
#配置 log 文件地址
#默认值为 stdout,标准输出,若后台模式会输出到 /dev/null
logfile /var/log/redis/redis.log
#可用数据库数
#默认值为 16,默认数据库为 0,数据库范围在 0~( database-1 )之间
databases 16
#数据写入磁盘快照设置
#保存数据到磁盘,格式如下
#save <seconds><changes>
#指出在多长时间内,有多少次更新操作,就将数据同步到数据文件 rdb
#相当于条件触发抓取快照,这个可以多个条件配合
#比如默认配置文件中的设置,就设置了 3 个条件
#save 900 1 : 900s 内至少有 1 个 key 被改变
#save 300 10 : 300s 内至少有 300 个 key 被改变
#save 60 10000 : 60s 内至少有 10000 个 key 被改变
# save 900 1
# save 300 10
# save 60 10000
# 后台存储错误停止写
stop-writes-on-bgsave-error yes
# 存储至本地数据库时(持久化到 rdb 文件)是否压缩数据,默认为 yes
rdbcompression yes
# 本地持久化数据库文件名,默认值为 dump.rdb
dbfilename dump.rdb
# 工作目录
# 数据库镜像备份的文件放置的路径
# 这里的路径跟文件名要分开配置是因为 Redis 在进行备份时,先会将当前数据库的状态写入到一
# 个临时文件中,等备份完成
```

```
# 再把该该临时文件替换为上面所指定的文件,而这里的临时文件和上面所配置的备份文件都会放
# 在这个指定的路径当中
# AOF 文件也会存放在这个目录下面
# 注意这里必须指定一个目录而不是文件
dir /var/lib/redis/
########################## 复制 ##########################
# 主从复制,设置该数据库为其他数据库的从数据库
# 设置当本机为 slave 服务时,设置 master 服务的 IP 地址及端口,在 Redis 启动时,它会自动从
# master 进行数据同步
# slaveof <masterip><masterport>
# 当 master 服务设置了密码保护时 ( 用 requirepass 制定的密码 )
# slave 服务连接 master 的密码
# masterauth <master-password>
# 当从库同主机失去连接或者复制正在进行,从机库有两种运行方式
# (1) 如果 slave-serve-stale-data 设置为 yes( 默认设置 ),从库会继续响应客户端的请求
# (2) 如果 slave-serve-stale-data 设置为 no,除去 INFO 和 SLAVOF 命令之外的任何请求都会
# 返回一个
# 错误 "SYNC with master in progress"
slave-serve-stale-data yes
# 配置 slave 实例是否接受写.写 slave 对存储短暂数据(在同 master 数据同步后可以很容易地被
# 删除)是有用的,但未配置的情况下,客户端写可能会发送问题
# 从 Redis 2.6 后,默认 slave 为 read-only
slaveread-only yes
# 从库会按照一个时间间隔向主库发送 pings,可以通过 repl-ping-slave-period 设置这个时间
# 间隔,默认是 10s
# repl-ping-slave-period 10
# repl-timeout: 设置主库批量数据传输时间或者 ping 回复时间间隔,默认值是 60s
# 一定要确保 repl-timeout 大于 repl-ping-slave-period
# repl-timeout 60
# 在 slave socket 的 SYNC 后禁用 TCP_NODELAY
# 如果选择" yes ",Redis 将使用一个较小的数字 TCP 数据包和更少的带宽将数据发送到 slave,但
# 是这可能导致数据发送到 slave 端会有延迟, 如果是 Linux kernel 的默认配置,会达到 40ms
# 如果选择 "no",则发送数据到 slave 端的延迟会降低,但将使用更多的带宽用于复制
repl-disable-tcp-nodelay no
# 设置复制的后台日志大小
# 复制的后台日志越大,slave 断开连接及后来可能执行部分复制花的时间就越长
# 后台日志在至少有一个 slave 连接时,仅仅分配一次
# repl-backlog-size 1mb
# 在 master 不再连接 slave 后,后台日志将被释放.下面的配置定义从最后一个 slave 断开连接后
# 需要释放的时间(s)
# 0 意味着从不释放后台日志
# repl-backlog-ttl 3600
# 如果 master 不能再正常工作,那么会在多个 slave 中,选择优先值最小的一个 slave 提升为
# master,优先值为 0 表示不能提升为 master
slave-priority 100
# 如果少于 N 个 slave 连接,且延迟时间≤Ms,则 master 可配置停止接受写操作
# 例如需要至少 3 个 slave 连接,且延迟≤10s 的配置
```

```
# min-slaves-to-write 3
# min-slaves-max-lag 10
# 设置 0 为禁用
# 默认 min-slaves-to-write 为 0 (禁用),min-slaves-max-lag 为 10
########################## 安全 ###########################
# 设置客户端连接后进行任何其他指定前需要使用的密码
# 警告: 因为 Redis 速度相当快,所以在一台比较好的服务器下,一个外部的用户可以在一秒钟进行
# 150K 次的密码尝试,这意味着你需要制定非常非常强大的密码来防止暴力破解
# requirepass jfedu
# 命令重命名
# 在一个共享环境下可以重命名相对危险的命令.例如把 CONFIG 重命名为一个不容易猜测的字符
# 举例
# rename-command CONFIG b840fc02d524045429941cc15f59e41cb7be6c52
# 如果想删除一个命令,直接把它重命名为一个空字符 "" 即可
# rename-command CONFIG ""
########################## 约束 ###########################
#设置同一时间最大客户端连接数,默认无限制
#Redis 可以同时打开的客户端连接数为 Redis 进程可以打开的最大文件描述符数
#如果设置 maxclients 0,表示不作限制
#当客户端连接数到达限制时,Redis 会关闭新的连接并向客户端返回 max number of clients
# reached 错误信息
# maxclients 10000
# 指定 Redis 最大内存限制,Redis 在启动时会把数据加载到内存中,达到最大内存后,Redis 会按
# 照清除策略尝试清除已到期的 key
# 如果 Redis 依照策略清除后无法提供足够空间,或者策略设置为"noeviction",则使用更多空间的
# 命令将会报错,例如 SET, LPUSH 等.但仍然可以进行读取操作
# 注意: Redis 新的 vm 机制,会把 key 存放内存,value 会存放在 swap 区
# 该选项对 LRU 策略很有用
# maxmemory 的设置比较适合于把 Redis 当作于类似 Memcached 的缓存来使用,而不适合当作一个
# 真实的 DB
# 当把 Redis 当作一个真实的数据库使用的时候,内存使用将是一个很大的开销
# maxmemory <bytes>
# 当内存达到最大值的时候 Redis 会选择删除哪些数据?有 5 种方式可供选择
# volatile-lru ->: 利用 LRU 算法移除设置过期时间的 key,LRU 即最近使用 (least recently
# used)
# allkeys-lru ->: 利用 LRU 算法移除任何 key
# volatile-random ->: 移除设置过期时间的随机 key
# allkeys->random ->移除随机 key,任何 key
# volatile-ttl ->: 移除即将过期的 key(minor TTL)
# noeviction ->: 不移除任何 key,只是返回一个写错误
# 注意: 对于上面的策略,如果没有合适的 key 可以移除,当写的时候 Redis 会返回一个错误
# 默认是 volatile-lru
# maxmemory-policy volatile-lru
# LRU 和 minimal TTL 算法都不是精准的算法,但是相对精确的算法 ( 为了节省内存 ),你可以随意
# 选择样本大小进行检测
# Redis 默认的会选择 3 个样本进行检测,你可以通过 maxmemory-samples 进行设置
# maxmemory-samples 3
```

```
############################ AOF ############################
# 默认情况下,Redis 会在后台异步的把数据库镜像备份到磁盘,但是该备份是非常耗时的,而且备
# 份也不能很频繁,如果发生诸如拉闸限电、拔插头等状况,那么将造成比较大范围的数据丢失
# 所以 Redis 提供了另外一种更加高效的数据库备份及灾难恢复方式
# 开启 append only 模式之后,Redis 会把所接收到的每一次写操作请求都追加到 appendonly.aof
# 文件中,当 Redis 重新启动时,会从该文件恢复出之前的状态
# 但是这样会造成 appendonly.aof 文件过大,所以 Redis 还支持了 BGREWRITEAOF 指令,对
# appendonly.aof 进行重新整理
# 你可以同时开启 asynchronous dumps 和 AOF
appendonly no
# AOF 文件名称 ( 默认为 "appendonly.aof")
# appendfilename appendonly.aof
# Redis 支持 3 种同步 AOF 文件的策略
# no: 不进行同步,系统去操作. Faster
# always: always 表示每次有写操作都进行同步 . Slow, Safest
# everysec: 表示对写操作进行累积,每秒同步一次 . Compromise
# 默认是 "everysec",按照速度和安全折中这是最好的
# 如果想让 Redis 能更高效的运行,你也可以设置为 "no",让操作系统决定什么时候去执行
# 或者相反想让数据更安全你也可以设置为 "always"
# 如果不确定就用 "everysec"
# appendfsync always
appendfsync everysec
# appendfsync no
# AOF 策略设置为 always 或者 everysec 时,后台处理进程 ( 后台保存或者 AOF 日志重写 ) 会执行
# 大量的 I/O 操作
# 在某些 Linux 配置中会阻止过长的 fsync() 请求.注意现在没有任何修复,即使 fsync 在另外一
# 个线程进行处理
# 为了减缓这个问题,可以设置下面这个参数 no-appendfsync-on-rewrite
no-appendfsync-on-rewrite no
# AOF 自动重写
# 当 AOF 文件增长到一定大小的时候 Redis 能够调用 BGREWRITEAOF 对日志文件进行重写
# 它是这样工作的: Redis 会记住上次进行写日志后文件的大小 ( 如果从开机以来还没进行过重
# 写,那日志大小在开机的时候确定 )
# 基础大小会同现在的大小进行比较.如果现在的大小比基础大小更大,重写功能将
# 启动
# 同时需要指定一个最小大小用于 AOF 重写,这个用于阻止即使文件很小但是增长幅度很大也去重
# 写 AOF 文件的情况
# 设置 percentage 为 0 就关闭这个特性
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
######################### LUA SCRIPTING #########################
# 一个 LUA 脚本最长的执行时间为 5000ms( 5s),如果为 0 或负数表示无限执行时间
lua-time-limit 5000
########################### LOW LOG ###########################
# Redis slow log: 记录超过特定执行时间的命令.执行时间不包括 I/O 计算,例如连接客户端,返回
# 结果等,只是命令执行时间
# 可以通过两个参数设置 slow log.一个是告诉 Redis 执行超过多少时间被记录的参数 slowlog-
```

```
# log - slower - than( μs )
# 另一个是 slow log 的长度. 当一个新命令被记录的时候最早的命令将被从队列中移除
# 下面的时间以 μs 为单位,因此 1000000 代表 1s
# 注意指定一个负数将关闭慢日志,而设置为 0 将强制每个命令都会记录
slowlog - log - slower - than 10000
# 对日志长度没有限制,只是要注意它会消耗内存
# 可以通过 SLOWLOG RESET 回收被慢日志消耗的内存
# 推荐使用默认值 128,当慢日志超过 128 时,最先进入队列的记录会被踢出
slowlog - max - len 128
```

12.7 Redis 常用配置

Redis 缓存服务器命令行中常用命令如下：

Redis CONFIG 命令格式如下：

```
redis 127.0.0.1:6379 > CONFIG GET|SET CONFIG_SETTING_NAME
```

详解如下：

- CONFIG GET *：获取 Redis 服务器所有配置信息。
- CONFIG SET loglevel "notice"：设置 Redis 服务器日志级别。
- CONFIG SET requirepass "jfedu"：配置 Redis 访问密码。
- AUTH jfedu：登录 Redis，执行 AUTH jfedu。
- redis-cli -h host -p port -a password：远程连接 Redis 数据库。
- CLIENT GETNAME：获取连接的名称。
- CLIENT SETNAME：设置当前连接的名称。
- CLUSTER SLOTS：获取集群节点的映射数组。
- COMMAND：获取 Redis 命令详情数组。
- COMMAND COUNT：获取 Redis 命令总数。
- COMMAND GETKEYS：获取给定命令的所有键。
- TIME：返回当前服务器时间。
- CONFIG GET parameter：获取指定配置参数的值。
- CONFIG SET parameter value：修改 Redis 配置参数，无须重启。
- CONFIG RESETSTAT：重置 INFO 命令中的某些统计数据。
- DBSIZE：返回当前数据库的 key 的数量。
- DEBUG OBJECT key：获取 key 的调试信息。
- DEBUG SEGFAULT：让 Redis 服务崩溃。
- FLUSHALL：删除所有数据库的所有 key。
- FLUSHDB：删除当前数据库的所有 key。
- ROLE：返回主从实例所属的角色。

- SAVE：异步保存数据到硬盘。
- SHUTDOWN：异步保存数据到硬盘，并关闭服务器。
- SLOWLOG：管理 Redis 的慢日志。
- SET keys values：设置 key 和 value。
- DEL jfedu：删除 key 及值。
- INFO CPU：查看服务器 CPU 占用信息。
- KEYS jfedu：查看是存在 jfedu 的 key。
- KEYS *：查看 Redis 所有的 key。
- CONFIG REWRITE：启动 Redis 时所指定的 redis.conf 配置文件进行改写。
- INFO [section]：获取 Redis 服务器的各种信息和统计数值。
- SYNC：用于复制功能(replication)的内部命令。
- SLAVEOF host port：指定服务器的从服务器(slave server)。
- MONITOR：实时打印出 Redis 服务器接收到的命令，调试用。
- LASTSAVE：返回最近一次 Redis 成功将数据保存到磁盘上的时间。
- CLIENT PAUSEtimeout：指定时间内终止运行来自客户端的命令。
- BGREWRITEAOF：异步执行一个 AOF(append only file)文件重写操作。
- BGSAVE：后台异步保存当前数据库的数据到磁盘。

12.8 Redis 集群主从实战

为了提升 Redis 高可用性，除了 Redis dump 数据之外，还需要配置 Redis 主从架构，可以利用主从架构将数据库持久化(数据持久化通俗讲就是把数据保存到磁盘上，保证不会因为断电等因素丢失数据)。

Redis 需要经常将内存中的数据同步到磁盘来保证持久化。Redis 支持两种持久化方式：一种是 snapshotting(快照)；另一种是 append only file(AOF)的方式。

Redis 主从复制，当用户往 master 端写入数据时，通过 Redis sync 机制将数据文件发送至 slave，slave 也会执行相同的操作确保数据一致。

Redis 主从配置非常简单，只需要在 Redis 从库 192.168.149.130 配置中设置如下指令，slaveof 表示指定主库的 IP，192.168.149.129 为 master 服务器，6379 为 master 服务器 Redis 端口，配置方法如下：

(1) 192.168.149.129 Redis 主库 redis.conf 配置文件如下：

```
daemonize no
pidfile /var/run/redis.pid
port 6379
tcp - backlog 511
timeout 0
tcp - keepalive 0
```

```
loglevel notice
logfile ""
databases 16
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename redis.rdb
dir /data/redis/
slave-serve-stale-data yes
slave-read-only yes
repl-disable-tcp-nodelay no
slave-priority 100
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64MB
lua-time-limit 5000
slowlog-log-slower-than 10000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
hash-max-ziplist-entries 512
hash-max-ziplist-value 64
list-max-ziplist-entries 512
list-max-ziplist-value 64
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
activerehashing yes
client-output-buffer-limit normal 0 0 0
client-output-buffer-limit slave 256MB 64MB 60
client-output-buffer-limit pubsub 32MB 8MB 60
hz 10
aof-rewrite-incremental-fsync yes
```

（2）192.168.149.130 Redis 从库 redis.conf 配置文件如下：

```
daemonize no
pidfile /var/run/redis.pid
port 6379
```

```
slaveof 192.168.149.129 6379
tcp-backlog 511
timeout 0
tcp-keepalive 0
loglevel notice
logfile ""
databases 16
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename redis.rdb
dir /data/redis/
slave-serve-stale-data yes
slave-read-only yes
repl-disable-tcp-nodelay no
slave-priority 100
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64MB
lua-time-limit 5000
slowlog-log-slower-than 10000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
hash-max-ziplist-entries 512
hash-max-ziplist-value 64
list-max-ziplist-entries 512
list-max-ziplist-value 64
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
activerehashing yes
client-output-buffer-limit normal 0 0 0
client-output-buffer-limit slave 256MB 64MB 60
client-output-buffer-limit pubsub 32MB 8MB 60
hz 10
aof-rewrite-incremental-fsync yes
```

（3）重启 Redis 主库、从库服务，在 Redis 主库创建 key 及 values，登录 Redis 从库查看，如图 12-14 所示。

```
[root@www-jfedu-net-129 ~]# redis-cli
redis 127.0.0.1:6379>
redis 127.0.0.1:6379> set jf1  www.jf1.com
OK
redis 127.0.0.1:6379>
redis 127.0.0.1:6379> set jf2  www.jf2.com
OK
redis 127.0.0.1:6379>
redis 127.0.0.1:6379> set jf3  www.jf3.com
OK
redis 127.0.0.1:6379>
redis 127.0.0.1:6379> get jf1
"www.jf1.com"
redis 127.0.0.1:6379>
redis 127.0.0.1:6379> get jf2
"www.jf2.com"
```

(a) Redis主库创建key

```
[root@192-168-149-130-jfedu ~]# redis-cli
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> get jf1
"www.jf1.com"
127.0.0.1:6379>
127.0.0.1:6379> get jf2
"www.jf2.com"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> get jf3
"www.jf3.com"
127.0.0.1:6379>
```

(b) Redis从库获取key值

图 12-14　Redis 主库、从库服务

12.9　Redis 数据备份与恢复

Redis 所有数据都是保存在内存中。Redis 数据备份可以定期地通过异步方式保存到磁盘上，该方式称为半持久化模式，如果每一次数据变化都写入 AOF 文件里面，则称为全持久化模式。同时还可以基于 Redis 主从复制实现 Redis 备份与恢复。

12.9.1　半持久化 RDB 模式

半持久化 RDB 模式也是 Redis 备份默认方式，是通过快照（snapshotting）完成的，当满足在 redis.conf 配置文件中设置的条件时，Redis 会自动将内存中的所有数据进行快照并存储在硬盘上，完成数据备份。

Redis 进行 RDB 快照的条件由用户在配置文件中自定义，由两个参数构成：时间和改动的键的个数。当在指定的时间内被更改的键的个数大于指定的数值时就会进行快照。在配置文件中已经预置了以下 3 个条件：

- save 900 1：900s 内有至少 1 个键被更改则进行快照。
- save 300 10：300s 内有至少 10 个键被更改则进行快照。
- save 60　10000：60s 内有至少 10000 个键被更改则进行快照。

默认可以存在多个条件，条件之间是或的关系，只要满足其中一个条件，就会进行快照。

如果想禁用自动快照，只需要将所有的 save 参数删除即可。Redis 默认会将快照文件存储在 Redis 数据目录，默认文件名为 dump. rdb 文件，可以通过配置 dir 和 dbfilename 两个参数分别指定快照文件的存储路径和文件名。也可以在 Redis 命令行执行 config get dir 获取 Redis 数据保存路径，如图 12-15 所示。

```
111) "slaveof"
112) ""
113) "notify-keyspace-events"
114) ""
115) "bind"
116) ""
redis 127.0.0.1:6379> config get dir
1) "dir"
2) "/data/redis"
redis 127.0.0.1:6379>
redis 127.0.0.1:6379>
redis 127.0.0.1:6379>
redis 127.0.0.1:6379>
redis 127.0.0.1:6379>
```

(a) 获取Redis数据目录

```
[root@www-jfedu-net-129 ~]#
[root@www-jfedu-net-129 ~]# ps -ef |grep redis
root      1381  1124  0 09:28 pts/0    00:00:00 /usr/local/redis/bin/re
root      1393  1124  0 09:28 pts/0    00:00:00 grep redis
[root@www-jfedu-net-129 ~]#
[root@www-jfedu-net-129 ~]# cd /data/redis/
[root@www-jfedu-net-129 redis]#
[root@www-jfedu-net-129 redis]# ls
dump.rdb
[root@www-jfedu-net-129 redis]#
[root@www-jfedu-net-129 redis]# ll
total 4
-rw-r--r-- 1 root root 18 Jun 10 09:28 dump.rdb
[root@www-jfedu-net-129 redis]#
```

(b) Redis数据目录及dump.rdb文件

图 12-15　Redis 数据目录

Redis 实现快照的过程，Redis 使用 fork 函数复制一份当前进程（父进程）的副本（子进程），父进程继续接收并处理客户端发来的命令，而子进程开始将内存中的数据写入硬盘中的临时文件，当子进程写入完所有数据后会用该临时文件替换旧的 RDB 文件，至此一次快照操作完成。

执行 fork 时操作系统会使用写时复制（copy-on-write）策略，即 fork 函数发生的一刻父子进程共享同一内存数据，当父进程要更改其中某片数据时，操作系统会将该片数据复制一份以保证子进程的数据不受影响，所以新的 RDB 文件存储的是执行 fork 那一刻的内存数据。

Redis 在进行快照的过程中不会修改 RDB 文件，只有快照结束后才会将旧的文件替换成新的，也就是说任何时候 RDB 文件都是完整的。这使得用户可以通过定时备份 RDB 文件来实现 Redis 数据库备份。

RDB 文件是经过压缩（可以配置 rdbcompression 参数以禁用压缩节省 CPU 占用）的二进制格式，所以占用的空间会小于内存中的数据大小，更加利于传输。除了自动快照，还可以手动发送 save 和 bgsave 命令让 Redis 执行快照，两个命令的区别在于，前者是由主进程

进行快照操作，会阻塞住其他请求，后者会通过 fork 子进程进行快照操作。

Redis 启动后会读取 RDB 快照文件，将数据从硬盘载入到内存，根据数据量大小、结构和服务器性能不同，通常将记录一千万个字符串类型键、大小为 1GB 的快照文件载入到内存中需花费 20～30s。

通过 RDB 方式实现持久化，一旦 Redis 异常退出，就会丢失最后一次快照以后更改的所有数据。此时需要开发者根据具体的应用场合，通过组合设置自动快照条件的方式来将可能发生的数据损失控制在能够接受的范围内。

12.9.2 全持久化 AOF 模式

如果数据很重要无法承受任何损失，可以考虑使用 AOF 方式进行持久化，默认 Redis 没有开启 AOF 方式的全持久化模式。

在启动时 Redis 会逐个执行 AOF 文件中的命令来将硬盘中的数据载入到内存中，载入的速度相较 RDB 会慢一些，开启 AOF 持久化后每执行一条会更改 Redis 中的数据的命令，Redis 就会将该命令写入硬盘中的 AOF 文件。AOF 文件的保存位置和 RDB 文件的位置相同，都是通过 dir 参数设置的，默认的文件名为 appendonly.aof，可以通过 appendfilename 参数修改该名称。

Redis 允许同时开启 AOF 和 RDB，既保证了数据安全又使得进行备份等操作十分容易。此时重新启动 Redis 后 Redis 会使用 AOF 文件来恢复数据，因为 AOF 方式的持久化可能丢失的数据更少，可以在 redis.conf 中通过 appendonly 参数开启 Redis AOF 全持久化模式，代码如下：

```
appendonly yes
appendfilename appendonly.aof
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64MB
appendfsync always
#appendfsync everysec
#appendfsync no
```

Redis AOF 持久化参数配置详解如下：

- appendonly yes：开启 AOF 持久化功能。
- appendfilename appendonly.aof：AOF 持久化保存文件名。
- appendfsync always：每次执行写入都会执行同步，最安全也最慢。
- #appendfsync everysec：每秒执行一次同步操作。
- #appendfsync no：不主动进行同步操作，而是完全交由操作系统来做，每 30s 一次，最快也最不安全。
- auto-aof-rewrite-percentage 100：当 AOF 文件大小超过上一次重写时的 AOF 文件大小的百分之多少时会再次进行重写，如果之前没有重写过，则以启动时的 AOF 文件大小为依据。

- auto-aof-rewrite-min-size 64MB：允许重写的最小 AOF 文件大小配置写入 AOF 文件后，要求系统刷新硬盘缓存的机制。

12.9.3 Redis 主从复制备份

通过持久化功能，Redis 保证了即使在服务器重启的情况下也不会损失（或少量损失）数据。但是由于数据是存储在一台服务器上的，如果这台服务器的硬盘出现故障，也会导致数据丢失。

为了避免单点故障，希望将数据库复制多个副本以部署在不同的服务器上，即使有一台服务器出现故障其他服务器依然可以继续提供服务，这就要求当一台服务器上的数据库更新后，可以自动将更新的数据同步到其他服务器上，Redis 提供了复制（replication）功能可以自动实现同步的过程。通过配置文件在 Redis 从数据库中配置文件中加入 slaveof master-ip master-port 即可，主数据库无须配置。

Redis 主从复制优点及应用场景，Web 应用程序可以基于主从同步实现读写分离以提高服务器的负载能力。在常见的场景中，读的频率一般比较大，当单机 Redis 无法应付大量的读请求时，可以通过复制功能建立多个从数据库，主数据库只进行写操作，而从数据库负责读操作，还可以基于 LVS＋keepalived 对 Redis 实现均衡和高可用。

从数据库持久化通常相对比较耗时，为了提高性能，可以通过复制功能建立一个（或若干个）从数据库，并在从数据库中启用持久化，同时在主数据库禁用持久化。

当从数据库崩溃时重启后主数据库会自动将数据同步过来，所以无须担心数据丢失。而当主数据库崩溃时，需要在从数据库中使用 slaveof no one 命令将从数据库提升成主数据库继续服务，并在原来的主数据库启动后使用 slaveof 命令将其设置成新的主数据库的从数据库，即可将数据同步回来。

12.10 LAMP 企业架构读写分离

LAMP＋Discuz＋Redis 缓解了 MySQL 的部分压力，但是如果访问量非常大，Redis 缓存中第一次没有缓存数据，会导致 MySQL 数据库压力增大，此时可以基于分库、分表、分布式集群或者读写分离来分担 MySQL 数据库的压力，以读写分离为案例，来实现分担 MySQL 数据库的压力。

MySQL 读写分离的原理：让 master 数据库处理事务增加、删除、修改、更新操作（create、insert、update、delete），而让 slave 数据库处理 select 操作，MySQL 读写分离前提是基于 MySQL 主从复制，这样可以保证在 master 上修改数据，slave 同步之后，Web 应用可以读取到 slave 端的数据。

实现 MySQL 读写分离可以基于第三方插件，也可以通过开发修改代码实现，具体实现读写分离的常见方式有以下 4 种：

- MySQL proxy 读写分离；

- Amoeba 读写分离；
- Mycat 读写分离；
- 基于程序读写分离(效率很高，但实施难度大，需开发改代码)。

Amoeba 是以 MySQL 为底层数据存储，并对 Web、App 应用提供 MySQL 协议接口的 proxy。它集中地响应 Web 应用的请求，依据用户事先设置的规则，将 SQL 请求发送到特定的数据库上执行，基于此可以实现负载均衡、读写分离、高可用性等需求。

Amoeba 相当于一个 SQL 请求的路由器，目的是为负载均衡、读写分离、高可用性提供机制，而不是完全实现它们。用户需要结合使用 MySQL 的 replication 等机制来实现副本同步等功能。

MySQL proxy 是 MySQL 官方提供的 MySQL 中间件服务，支持无数客户端连接，同时后端可连接若干台 MySQL server 服务器，MySQL proxy 自身基于 MySQL 协议，连接 MySQL proxy 的客户端无须修改任何设置，跟正常连接 MySQL server 没有区别，无须修改程序代码。

MySQL proxy 是 App 应用(客户端)与 MySQL server 之间的一个连接代理，MySQL proxy 负责将 App 应用的 SQL 请求根据转发规则，转发至相应的后端数据库，基于 LUA 脚本，可以实现复杂的连接控制和过滤，从而实现数据读写分离和负载均衡的需求。

MySQL proxy 允许用户指定 LUA 脚本对 SQL 请求进行拦截，对请求进行分析与修改，还允许用户指定 LUA 脚本对服务器的返回结果进行修改，加入一些结果集或者去除一些结果集，对 SQL 的请求通常为读请求、写请求，基于 LUA 脚本，可以实现将 SQL 读请求转发至后端 slave 服务器，将 SQL 写请求转发至后端 master 服务器。

如图 12-16 所示，为 MySQL proxy 读写分离架构图，通过架构图可以清晰地看到 SQL 请求整个流向的过程。

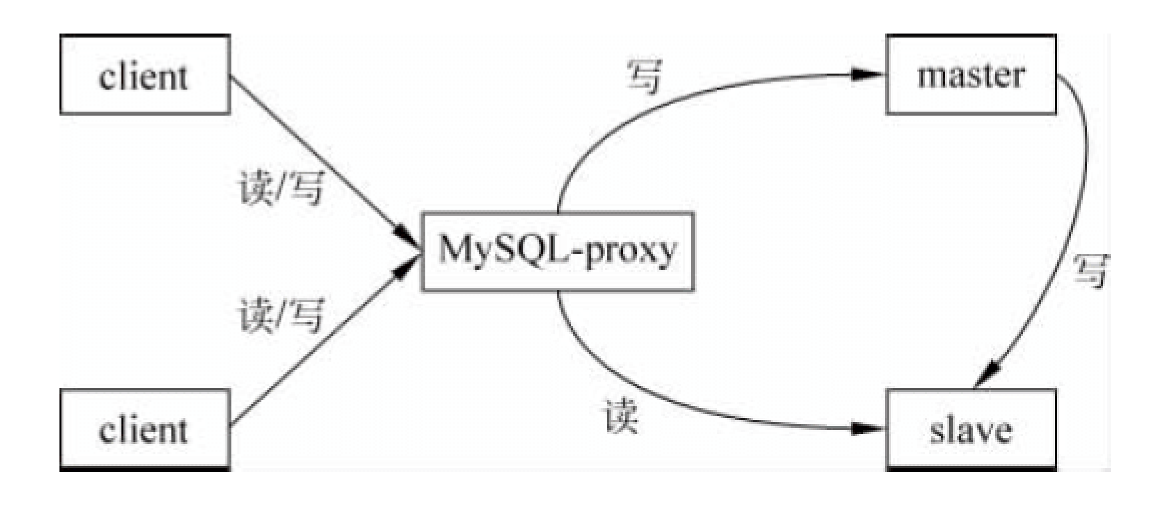


图 12-16 MySQL proxy 读写分离流程

MySQL proxy 读写分离架构实战配置，如图 12-17 所示，两台 Web 通过 MySQL proxy 连接后端 1.14 和 1.15 MySQL 服务器。

构建 MySQL 读写分离架构首先需要将两台 MySQL 服务器配置为主从复制(前文已提到，此处省略配置)，配置完毕后，在 192.168.1.16 服务器上安装 MySQL proxy 服务即可，配置步骤如下：

(1) 下载 MySQL proxy 软件版本，解压并重命名至/usr/local/mysql-proxy，命令

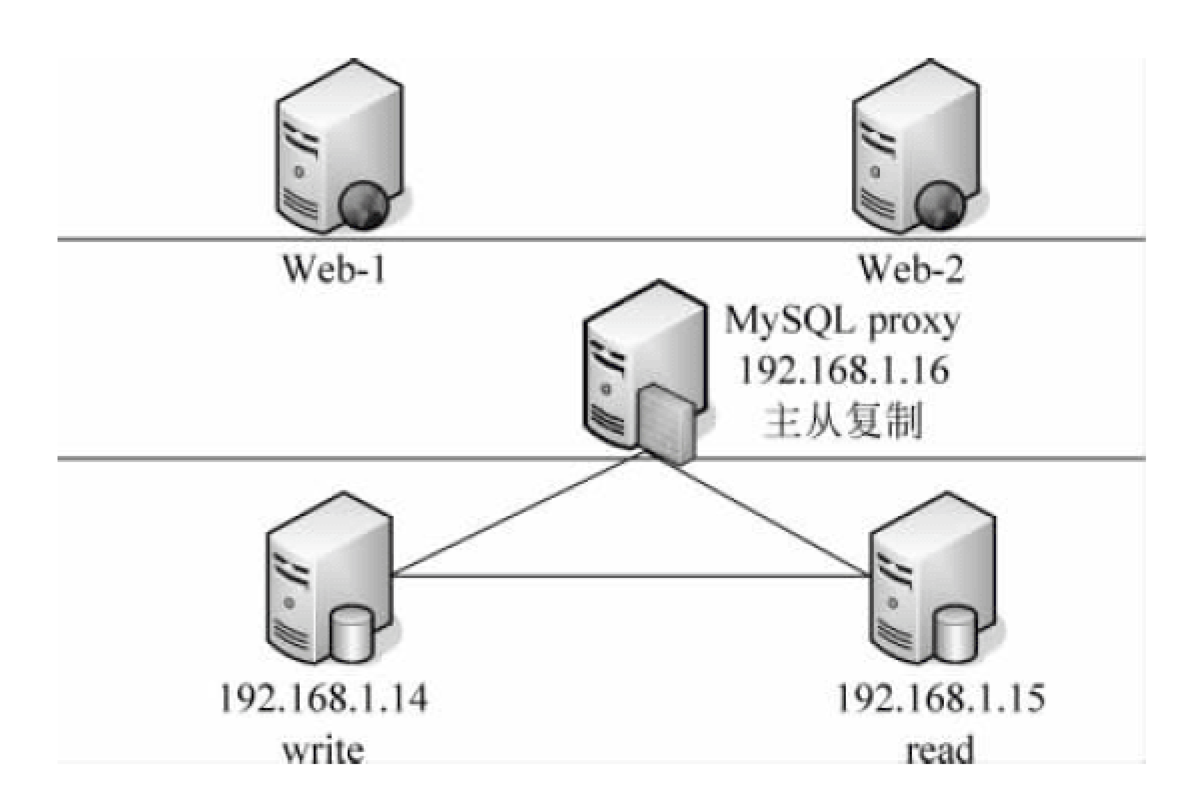


图 12-17　MySQL proxy 实施架构图

如下：

```
wget http://ftp.ntu.edu.tw/pub/MySQL/Downloads/MySQL-Proxy/mysql-proxy-0.8.4-linux-
el6-x86-64bit.tar.gz
useradd -r mysql-proxy
tar zxvf mysql-proxy-0.8.4-linux-el6-x86-64bit.tar.gz -C /usr/local
mv /usr/local/mysql-proxy-0.8.4-linux-el6-x86-64bit /usr/local/mysql-proxy
```

（2）在环境变量配置文件/etc/profile 中加入如下代码保存退出，然后执行 source /etc/profile 使配置生效即可，命令如下：

```
export PATH=$PATH:/usr/local/mysql-proxy/bin/
```

（3）启动 MySQL proxy 中间件，命令如下：

```
mysql-proxy --daemon --log-level=debug --user=mysql-proxy --keepalive
--log-file=/var/log/mysql-proxy.log --plugins="proxy"
--proxy-backend-addresses="192.168.1.14:3306"
--proxy-read-only-backend-addresses="192.168.1.15:3306"
--proxy-lua-script="/usr/local/mysql-proxy/share/doc/mysql-proxy/rw-splitting.lua"
--plugins=admin --admin-username="admin" --admin-password="admin"
--admin-lua-script="/usr/local/mysql-proxy/lib/mysql-proxy/lua/admin.lua"
```

（4）MySQL proxy 的相关参数详解如下：

- --help-all：获取全部帮助信息。
- --proxy-address=host:port：代理服务监听的地址和端口，默认为 4040。
- --admin-address=host:port：管理模块监听的地址和端口，默认为 4041。
- --proxy-backend-addresses=host:port：后端 MySQL 服务器的地址和端口。
- --proxy-read-only-backend-addresses=host:port：后端只读 MySQL 服务器的地址和端口。
- --proxy-lua-script=file_name：完成 MySQL 代理功能的 LUA 脚本。

- --daemon：以守护进程模式启动 MySQL proxy。
- --keepalive：在 MySQL proxy 崩溃时尝试重启。
- --log-file=/path/to/log_file_name：日志文件名称。
- --log-level=level：日志级别。
- --log-use-syslog：基于 syslog 记录日志。
- --plugins=plugin：在 MySQL proxy 启动时加载的插件。
- --user=user_name：运行 MySQL proxy 进程的用户。
- --defaults-file=/path/to/conf_file_name：默认使用的配置文件路径，其配置段使用[mysql-proxy]标识。
- --proxy-skip-profiling：禁用 profile。
- --pid-file=/path/to/pid_file_name：进程文件名。

(5) MySQL proxy 启动后，在服务器端查看端口，其中 4040 为 proxy 代理端口用于 Web 应用连接，4041 位管理端口用于 SA 或者 DBA 管理，如图 12-18 所示。

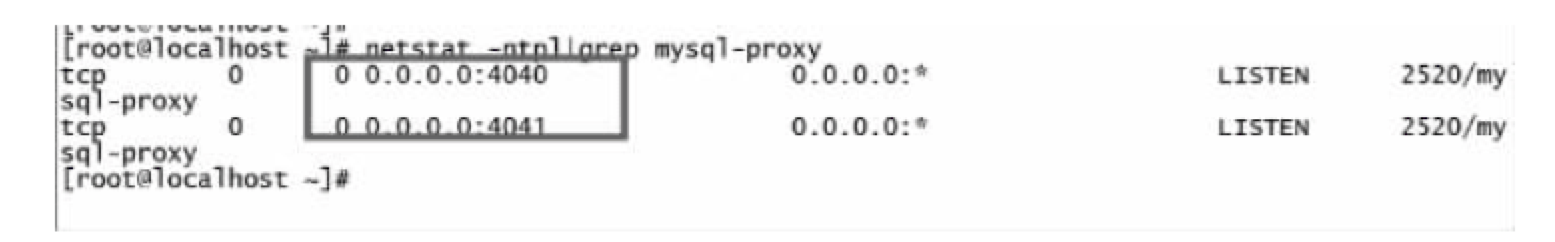


图 12-18 MySQL proxy 启动端口

(6) 基于 4041 端口 MySQL proxy 查看读写分离状态，登录 4041 管理端口，命令如下：

```
mysql -h192.168.1.16 -uadmin -p -P 4041
```

(7) 以 4041 管理口登录，然后执行 select 命令，命令如下，如图 12-19 所示 state 均为 up 状态，type 类型为 rw、ro，则证明读写分离状态成功。如果状态为 unknown 未知状态，以 4040 端口登录执行“show databases;”命令，直到 state 变成 up 状态为止。

```
select * from backends;
```

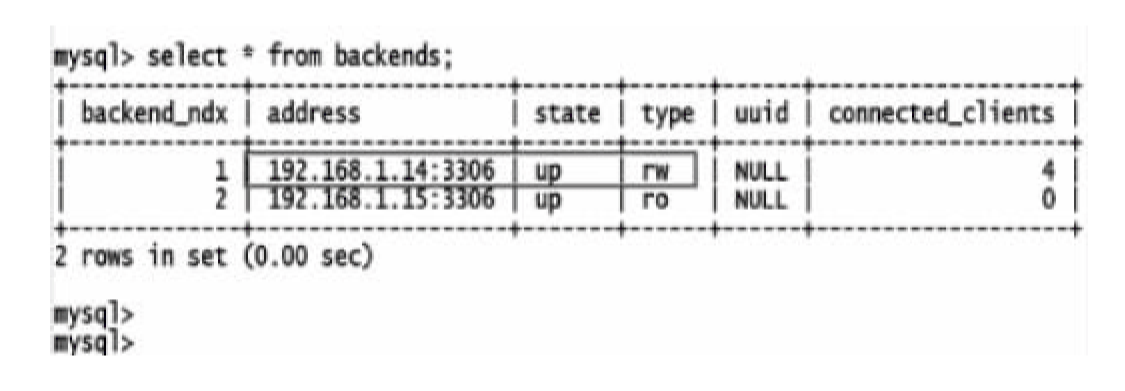


图 12-19 MySQL proxy 读写分离状态

(8) 读写分离数据测试，以 3306 端口登录到从库，进行数据写入和测试，在从库上创建 jfedu_test 测试库，并写入内容，如图 12-20 所示。

(9) 读写分离数据测试，以 4040 代理端口登录，执行如下命令，可以查看到数据即证明读写分离成功。

```
mysql> create database jfedu_test;
Query OK, 1 row affected (0.00 sec)

mysql>
mysql>
mysql> use jfedu_test
Database changed
mysql>
mysql> create table t1 (id char(20),name char(20));
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql>
mysql> insert into t1 values (01,'jfedu.net');
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 values (02,'jfteach.com');
Query OK, 1 row affected (0.00 sec)
```

图 12-20　MySQL proxy 读写分离测试

```
mysql -h192.168.1.16 -uroot -p123456 -P4040 -e "select * from jfedu_test.t1; "
```

（10）登录 Apache Web 服务器，修改 Discuz PHP 网站发布/usr/local/apache2/htdcos 目录，全局配置文件 config_global.php，查找 dbhost 段，将 192.168.1.16 改成 192.168.1.16:4040，如图 12-21 所示。

```
$_config = array();

// ---------------------------  CONFIG DB  ---------------------------
$_config['db']['1']['dbhost'] = '192.168.1.16:4040';
$_config['db']['1']['dbuser'] = 'root';
$_config['db']['1']['dbpw'] = '123456';
$_config['db']['1']['dbcharset'] = 'utf8';
$_config['db']['1']['pconnect'] = '0';
$_config['db']['1']['dbname'] = 'discuz';
$_config['db']['1']['tablepre'] = 'pre_';
$_config['db']['slave'] = '';
$_config['db']['common']['slave_except_table'] = '';

// -------------------------  CONFIG MEMORY  -------------------------
```

图 12-21　MySQL proxy 读写分离测试

第 13 章 Zabbix 分布式监控企业实战



企业服务器对用户提供服务，作为运维工程师最重要的责任就是保证该网站正常稳定的运行，需要实时监控网站、服务器的运行状态，并且出现故障及时处理。

监控网站无须人工时刻去访问 Web 网站或者登录服务器去检查，可以借助开源监控软件，例如 Zabbix、Cacti、Nagios、Ganglia 等来实现对网站的 7×24 小时的监控，并且做到有故障及时报警通知 SA 解决。

本章向读者介绍企业级分布式监控 Zabbix 入门、Zabbix 监控原理、最新版本 Zabbix 安装实战、Zabbix 批量监控客户端、监控 MySQL、Web 关键词及微信报警等内容。

13.1 Zabbix 监控系统入门简介

Zabbix 是一个基于 Web 界面的提供分布式系统监控的企业级开源解决方案，Zabbix 能监视各种网络参数，保证服务器系统安全稳定地运行，并提供灵活的通知机制以让 SA 快速定位并解决存在的各种问题。Zabbix 分布式监控系统的优点如下：

- 支持自动发现服务器和网络设备；
- 支持底层自动发现；
- 分布式的监控体系和集中式的 Web 管理；
- 支持主动监控和被动监控模式；
- 服务器端支持多种操作系统：Linux，Solaris，HP-UX，AIX，FreeBSD，OpenBSD，MAC 等；
- agent 客户端支持多种操作系统，如 Linux，Solaris，HP-UX，AIX，FreeBSD，Windows 等；
- 基于 SNMP、IPMI 接口、Zabbix Agent 方式监控客户端；
- 安全的用户认证及权限配置；
- 基于 Web 的管理方法，支持自由的自定义事件和邮件发送；
- 高水平的业务视图监控资源，支持日志审计、资产管理等功能；
- 支持高水平 API 二次开发、脚本监控、自 key 定义、自动化运维整合调用。

13.2 Zabbix 监控组件及流程

Zabbix 监控组件如图 13-1 所示，主要由三大部分组成：Zabbix server 端、Zabbix proxy、agent 客户端，其中 Zabbix server 端包括 Web GUI、database、Zabbix server。

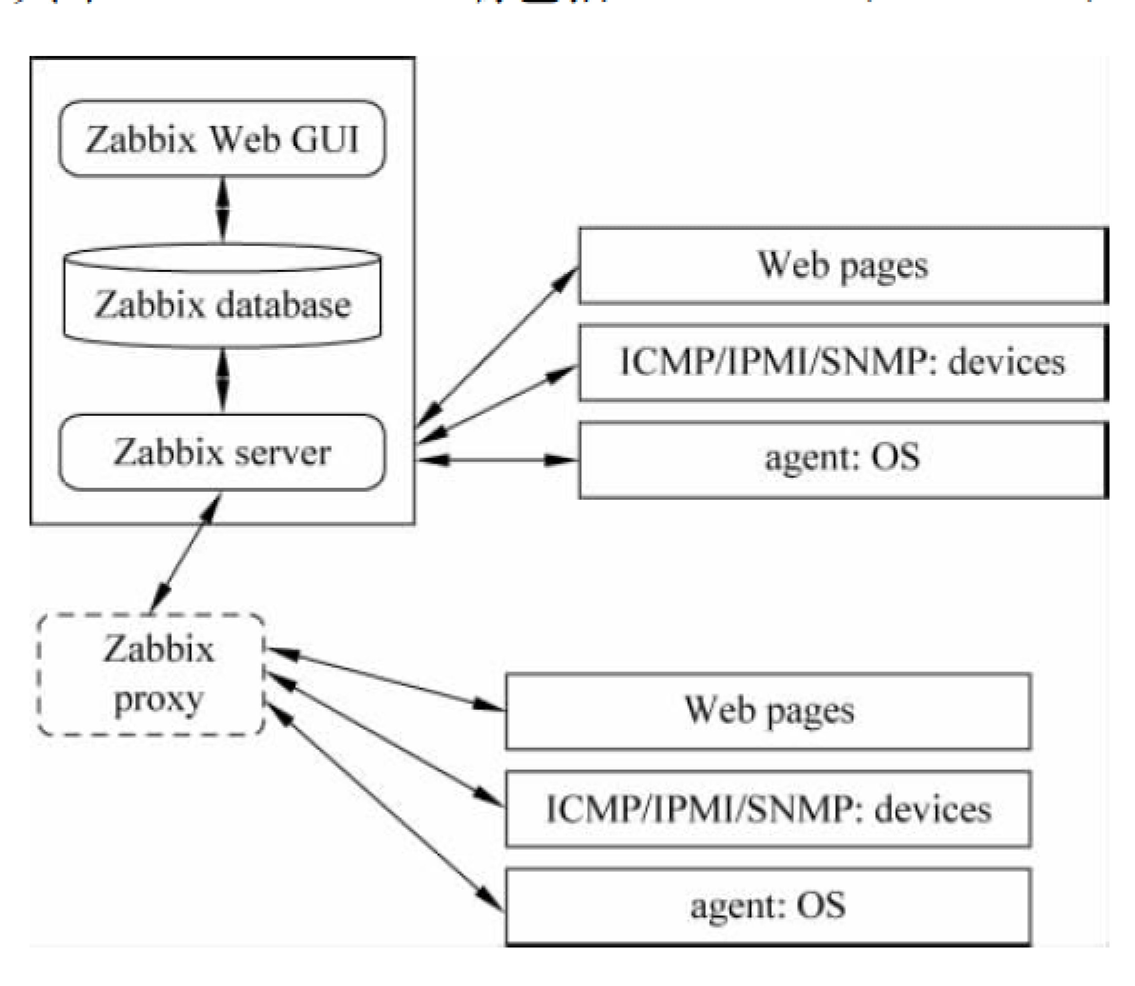


图 13-1　Zabbix 监控组件

Zabbix 监控系统的具体流程如图 13-2 所示。

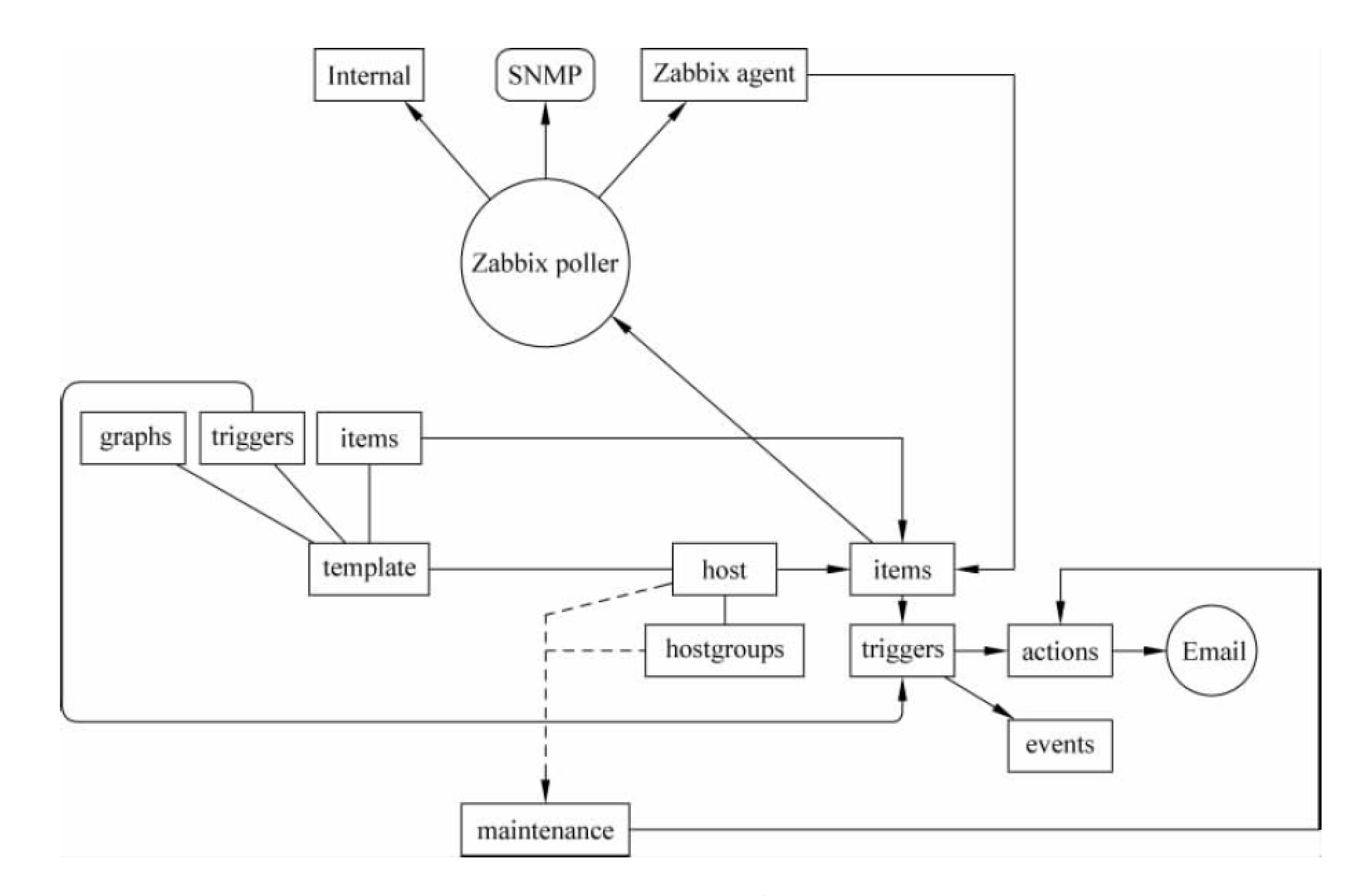


图 13-2　Zabbix 监控流程图

Zabbix 监控完整流程：Agent 安装在被监控的主机上，Agent 负责定期收集客户端本地各项数据，并发送到 Zabbix server 端，Zabbix server 收到数据，将数据存储到数据库中，用户基于 Zabbix Web 可以看到数据在前端展现的图像。

当 Zabbix 监控某个具体项目时，该项目会设置一个触发器阀值，当被监控的指标超过该触发器设定的阀值，会进行一些必要的动作，动作包括邮件、微信报警或者执行命令等操作。Zabbix 完整监控系统各个部分负责的工作如下：

- Zabbix server：负责接收 agent 发送的报告信息的核心组件，所有配置、统计数据及操作数据均由其组织进行。
- database storage：专用于存储所有配置信息以及存储由 Zabbix 收集到的数据。
- Web interface：Zabbix 的 GUI 接口，通常与 server 运行在同一台主机上。
- proxy：常用于分布监控环境中，代理 server 收集部分被监控端的监控数据并统一发往 server 端。
- Zabbix agent：部署在被监控主机上，负责收集本地数据并发往 server 端或 proxy 端。

Zabbix 监控部署在系统中，通常会包含 5 个常见的程序：zabbix_server、zabbix_get、zabbix_agentd、zabbix_proxy、zabbix_sender。5 个程序启动后分别对应 5 个进程，每个进程的功能如下：

- zabbix_server：Zabbix 服务端守护进程，其中 zabbix_agentd、zabbix_get、zabbix_sender、zabbix_proxy 的数据最终均是提交给 zabbix_server。
- zabbix_agentd：客户端守护进程，负责收集客户端数据，例如收集 CPU 负载、内存、硬盘使用情况等。
- zabbix_get：Zabbix 数据获取工具，单独使用的命令，通常在 server 或者 proxy 端执行获取远程客户端信息的命令。
- zabbix_sender：Zabbix 数据发送工具，用于发送数据给 server 或者 proxy，通常用于耗时比较长的检查，很多检查非常耗时间，导致 Zabbix 超时，于是需要在脚本执行完毕之后，使用 sender 主动提交数据。
- zabbix_proxy：Zabbix 分布式代理守护进程，分布式监控架构需要部署 zabbix_proxy。

13.3 Zabbix 监控方式及数据采集

Zabbix 分布式监控系统监控客户端的方式常见有 3 种：agent 方式、SNMP 方式、IPMI 方式，3 种方式特点如下：

- agent：Zabbix 可以基于自身 zabbix_agent 客户端插件监控 OS 的状态，例如 CPU、内存、硬盘、网卡、文件等。
- SNMP：Zabbix 可以通过简单网络管理协议（simple network management protocol，

SNMP)协议监控网络设备或者 Windows 主机等，通过设定 SNMP 的参数将相关监控数据传送至服务器端，交换机、防火墙等网络设备一般都支持 SNMP 协议。

- IPMI：智能平台管理接口(intelligent platform management interface，IPMI)即主要应用于设备的物理特性，包括温度、电压、电扇工作状态、电源供应以及机箱入侵等。IPMI 最大的优势在于无论 OS 在开机还是关机的状态下，只要接通电源就可以实现对服务器的监控。

Zabbix 监控客户端分为主动监控与被动监控，主被动模式以客户端为参照，Zabbix 监控客户端默认为被动模式，可以修改为主动模式，只需要在客户端配置文件中添加即可。关闭被动模式的方法为在配置文件中加入 StartAgents＝0，即为关闭被动模式。主被动监控模式区别如下：

- Zabbix 主动模式：agent 主动请求 server 获取主动的监控项列表，并主动将监控项内需要检测的数据提交给 server 或者 proxy，Zabbix agent 首先向 server active 配置的 IP 请求获取 active items，获取后将 active tiems 数据值提交给 server 或者 proxy。
- Zabbix 被动模式：server 向 agent 请求获取监控项的数据，agent 返回数据，server 打开一个 TCP 连接，server 发送请求 agent. ping，agent 接收到请求并且响应，server 处理接收到的数据。

13.4 Zabbix 监控概念

Zabbix 监控系统包括很多监控概念，掌握 Zabbix 监控概念有助于对 Zabbix 监控系统快速地理解，Zabbix 常用术语及解释如下：

- 主机(host)：被监控的网络设备，可以写 IP 或者 DNS。
- 主机组(host group)：主机组用于管理主机，可以批量设置权限。
- 监控项(item)：具体监控项，items 值有独立的 keys 进行识别。
- 触发器(trigger)：为某个 items 设置触发器，达到触发器会执行 action 动作。
- 事件(event)：例如达到某个触发器，称之为一个事件。
- 动作(action)：对于特定事件事先定义的处理方法，默认可以发送信息及发送命令。
- 报警升级(escalation)：发送警报或执行远程命令的自定义方案，如隔 5min 发送一次警报，共发送 5 次等。
- 媒介(media)：发送通知的方式，可以支持 mail、SMS、scripts 等。
- 通知(notification)：通过设置的媒介向用户发送的有关某事件的信息。
- 远程命令(remote command)：达到触发器，可以在被监控端执行命令。
- 模板(template)：可以快速监控被监控端，模块包含 item、trigger、graph、screen、application。
- Web 场景(Web scenario)：用于检测 Web 站点可用性，监控 HTTP 关键词。
- Web 前端(frontend)：Zabbix 的 Web 接口。

- 图形(graph)：监控图像。
- 屏幕(screens)：屏幕显示。
- 幻灯(slide show)：幻灯显示。

13.5 Zabbix 监控平台部署

Zabbix 监控平台部署，至少需要安装 4 个组件：Zabbix Server、Zabbix Web、databases、Zabbix Agent，以下为 Zabbix 监控平台安装配置详细步骤：

(1) 系统环境。

- server 端：192.168.149.128。
- agent 端：192.168.149.129。

(2) 下载 Zabbix 版本，各个版本之间安装方法相差不大，可以根据实际情况选择安装版本，本文安装版本为 zabbix-3.2.6.tar.gz。

```
wget http://sourceforge.net/projects/zabbix/files/ZABBIX%20Latest%20Stable/3.2.6/zabbix
-3.2.6.tar.gz/download
```

(3) Zabbix Server 端和 Zabbix Agent 执行如下代码：

```
yum -y install curl curl-devel net-snmp net-snmp-devel perl-DBI
groupadd zabbix ; useradd -g zabbix zabbix; usermod -s /sbin/nologin zabbix
```

(4) Zabbix Server 端配置。

创建 Zabbix 数据库，执行授权命令如下：

```
create database zabbix charset=utf8;
grant all on zabbix.* to zabbix@localhost identified by '123456';
flush privileges;
```

解压 Zabbix 软件包并将 Zabbix 基础 SQL 文件导入数据至 Zabbix 数据库，代码如下：

```
tar zxvf zabbix-3.2.6.tar.gz
cd zabbix-3.2.6
mysql -uzabbix -p123456 zabbix <database/mysql/schema.sql
mysql -uzabbix -p123456 zabbix <database/mysql/images.sql
mysql -uzabbix -p123456 zabbix <database/mysql/data.sql
```

切换至 Zabbix 解压目录，执行如下代码，安装 Zabbix Server。

```
./configure --prefix=/usr/local/zabbix/ --enable-server --enable-agent --with-
mysql --enable-ipv6 --with-net-snmp --with-libcurl
make
make install
ln -s /usr/local/zabbix/sbin/zabbix_* /usr/local/sbin/
```

Zabbix Server 安装完毕，cd /usr/local/zabbix/etc/目录，如图 13-3 所示。

```
[root@localhost etc]# ls
zabbix_agent.conf    zabbix_agentd.conf    zabbix_server.conf
zabbix_agent.conf.d  zabbix_agentd.conf.d  zabbix_server.conf.d
[root@localhost etc]# ll
total 24
-rw-r--r-- 1 root root 1601 May 19 21:52 zabbix_agent.conf
drwxr-xr-x 2 root root 4096 May 19 21:52 zabbix_agent.conf.d
-rw-r--r-- 1 root root  111 May 20 23:55 zabbix_agentd.conf
drwxr-xr-x 2 root root 4096 May 19 21:52 zabbix_agentd.conf.d
-rw-r--r-- 1 root root   94 May 20 23:55 zabbix_server.conf
drwxr-xr-x 2 root root 4096 May 19 21:53 zabbix_server.conf.d
[root@localhost etc]# pwd
/usr/local/zabbix/etc
[root@localhost etc]#
```

图 13-3 Zabbix 监控流程图

备份 Zabbix Server 配置文件，代码如下：

```
cp zabbix_server.conf zabbix_server.conf.bak
```

在 zabbix_server.conf 配置文件中设置代码如下：

```
LogFile = /tmp/zabbix_server.log
DBHost = localhost
DBName = zabbix
DBUser = zabbix
DBPassword = 123456
```

同时 cp zabbix_server 启动脚本至/etc/init.d/目录，启动 zabbix_server，zabbix_server 默认监听端口为 10051，代码如下：

```
cd zabbix-3.2.6
cp misc/init.d/tru64/zabbix_server /etc/init.d/zabbix_server
chmod o+x /etc/init.d/zabbix_server
```

配置 Zabbix interface Web 页面，安装 HTTP Web 服务器，将 Zabbix Web 代码发布至 Apache 默认发布目录，PHP 版本需要使用 PHP 5.4.0 以上版本，PHP 5.3 升级至 PHP 5.6 的步骤，代码如下：

```
rpm -Uvh http://repo.webtatic.com/yum/el6/latest.rpm
yum remove php*
yum install php56w.x86_64 php56w-cli.x86_64 php56w-common.x86_64 php56w-gd.x86_64 php56w
-ldap.x86_64 php56w-mbstring.x86_64 php56w-mcrypt.x86_64 php56w-mysql.x86_64 php56w-
pdo.x86_64 -y
yum install httpd httpd-devel httpd-tools -y
cp -a /root/zabbix-3.2.6/frontends/php/* /var/www/html/
sed -i '/date.timezone/i date.timezone = PRC' /etc/php.ini
```

重新启动 Zabbix Server、HTTP、MySQL 服务，代码如下：

```
/etc/init.d/zabbix_server restart
```

```
/etc/init.d/httpd restart
/etc/init.d/mysqld restart
```

（5）Zabbix Web GUI 安装配置。

通过浏览器 Zabbix Web 验证，浏览器访问 http://192.168.149.128/，如图 13-4 所示。

ZABBIX

Welcome
Check of pre-requisites
Configure DB connection
Zabbix server details
Pre-installation summary
Install

Welcome to

Zabbix 3.2

图 13-4　Zabbix Web 安装界面

单击“下一步”按钮，出现如图 13-5 所示的界面，如果有错误提示，需要把错误依次解决完，方可进行下一步操作。

ZABBIX

Welcome
Check of pre-requisites
Configure DB connection
Zabbix server details
Pre-installation summary
Install

Check of pre-requisites

– Minimum required size of PHP post is 16M (configuration option "post_max_size").
– Minimum required limit on input parse time for PHP scripts is 300 (configuration option "max_input_time").

	Current value	Required	
PHP version	5.6.30	5.4.0	OK
PHP option "memory_limit"	128M	128M	OK
PHP option "post_max_size"	8M	16M	Fail
PHP option "upload_max_filesize"	2M	2M	OK
PHP option "max_execution_time"	3000000	300	OK
PHP option "max_input_time"	60	300	Fail

图 13-5　Zabbix Web 安装错误提示

上述异常错误解决方法的代码如下，安装缺失的软包，并修改 php.ini 对应参数的值即可，如图 13-6 所示。

```
yum install php56w-mbstring php56w-bcmath php56w-gd php56w-xml -y
yum install gd gd-devel -y
sed -i '/post_max_size/s/8/16/g;/max_execution_time/s/30/300/g;/max_input_time/s/60/300/g;s/\;date.timezone.*/date.timezone \= PRC/g;s/\;always_populate_raw_post_data/always_populate_raw_post_data/g' /etc/php.ini
/etc/init.d/httpd restart
```

ZABBIX

Check of pre-requisites

Welcome
Check of pre-requisites
Configure DB connection
Zabbix server details
Pre-installation summary
Install

	Current value	Required	
PHP version	5.6.30	5.4.0	OK
PHP option "memory_limit"	128M	128M	OK
PHP option "post_max_size"	16M	16M	OK
PHP option "upload_max_filesize"	2M	2M	OK
PHP option "max_execution_time"	30000000	300	OK
PHP option "max_input_time"	300	300	OK
PHP option "date.timezone"	PRC		OK
PHP databases support	MySQL		OK
PHP bcmath	on		OK

图 13-6　Zabbix Web 测试安装环境

单击“下一步”按钮，如图 13-7 所示，配置数据库连接，输入数据库名、用户、密码，单击 Test connection，显示 OK，再单击“下一步”按钮即可。



图 13-7　Zabbix Web 数据库配置

继续单击“下一步”按钮，出现如图 13-8 所示的界面，填写 Zabbix Name 显示，可以为空，也可以输入自定义的名称。

ZABBIX

Zabbix server details

Please enter the host name or host IP address and port number of the Zabbix server, as well as the name of the installation (optional).

Welcome
Check of pre-requisites
Configure DB connection
Zabbix server details
Pre-installation summary
Install

Host: localhost
Port: 10051
Name: 京峰教育-分布式监控系统

图 13-8　Zabbix Web 详细信息

单击"下一步"按钮，如图 13-9 所示，需修创建 zabbix. conf. php 文件，单击 Download the configuration file 下载 zabbix. conf. php 文件，并将该文件上传至/var/www/html/conf/，同时设置可写权限，刷新 Web 页面，zabbix. conf. php 内容代码如下，最后单击 Finish 即可。

```
<?php
// Zabbix GUI configuration file
global $DB;
 $DB['TYPE']     = 'MYSQL';
 $DB['SERVER']   = 'localhost';
 $DB['PORT']     = '0';
 $DB['DATABASE'] = 'zabbix';
 $DB['USER']     = 'zabbix';
 $DB['PASSWORD'] = '123456';
// Schema name. Used for IBM DB2 and PostgreSQL.
 $DB['SCHEMA']   = '';
 $ZBX_SERVER     = 'localhost';
 $ZBX_SERVER_PORT = '10051';
 $ZBX_SERVER_NAME = '京峰教育-分布式监控系统';
 $IMAGE_FORMAT_DEFAULT = IMAGE_FORMAT_PNG;
```

图 13-9　Zabbix Web 配置文件测试

登录 Zabbix Web 界面，默认用户名和密码为 admin/zabbix，如图 13-10 所示。

（6）agent 客户端安装配置。

解压 zabbix-3.2.6.tar.gz 源码文件，切换至解压目录，编译安装 Zabbix，命令如下：

```
./configure --prefix=/usr/local/zabbix --enable-agent
make
make install
ln -s /usr/local/zabbix/sbin/zabbix_* /usr/local/sbin/
```

修改 zabbix_agentd. conf 客户端配置文件，指定 server IP，同时设置本地 Hostname 为本地 IP 地址或者 DNS 名称，命令如下：

```
LogFile=/tmp/zabbix_agentd.log
Server=192.168.149.128
```

(a) Zabbix Web登录界面

(b) Zabbix Web后台界面

图 13-10 Zabbix Web 界面

```
ServerActive = 192.168.149.128
Hostname = 192.168.149.129
```

同时执行命令 cp zabbix_agentd 启动脚本至/etc/init.d/目录，启动 zabbix_agentd 服务即可，zabbix_agentd 默认监听端口为 10050，命令如下：

```
cd zabbix - 3.2.6
cp misc/init.d/tru64/zabbix_agentd /etc/init.d/zabbix_agentd
chmod o + x /etc/init.d/zabbix_agentd
/etc/init.d/zabbix_agentd start
```

(7) Zabbix 监控客户端。

Zabbix 服务端和客户端安装完毕之后，需通过 Zabbix server 添加客户端监控，Zabbix Web 界面添加客户端监控的操作步骤如下：

依次选择 Zabbix-Web→configuration →hosts →Create host →Host name 和 Agent

interfaces，同时添加 templates 模板，选择 Add →Template OS Linux，并单击 Add 提交。此处 Host name 名称与 Agentd. conf 配置文件中 Hostname 保持一致，否则会报错，详情如图 13-11 所示。

Host name 192.168.149.129
Visible name
Groups In groups Other groups
Linux servers
Discovered hosts
Hypervisors
Templates
Virtual machines
Zabbix servers
New group
Agent interfaces IP address DNS name Connect to Port Default
192.168.149.129 IP DNS 10050 Remove

图 13-11　Zabbix 添加客户端监控

将客户端主机链接至 Template OS Linux，启用模板完成主机默认监控，单击 Add，再继续单击 Update 即可，如图 13-12 所示。

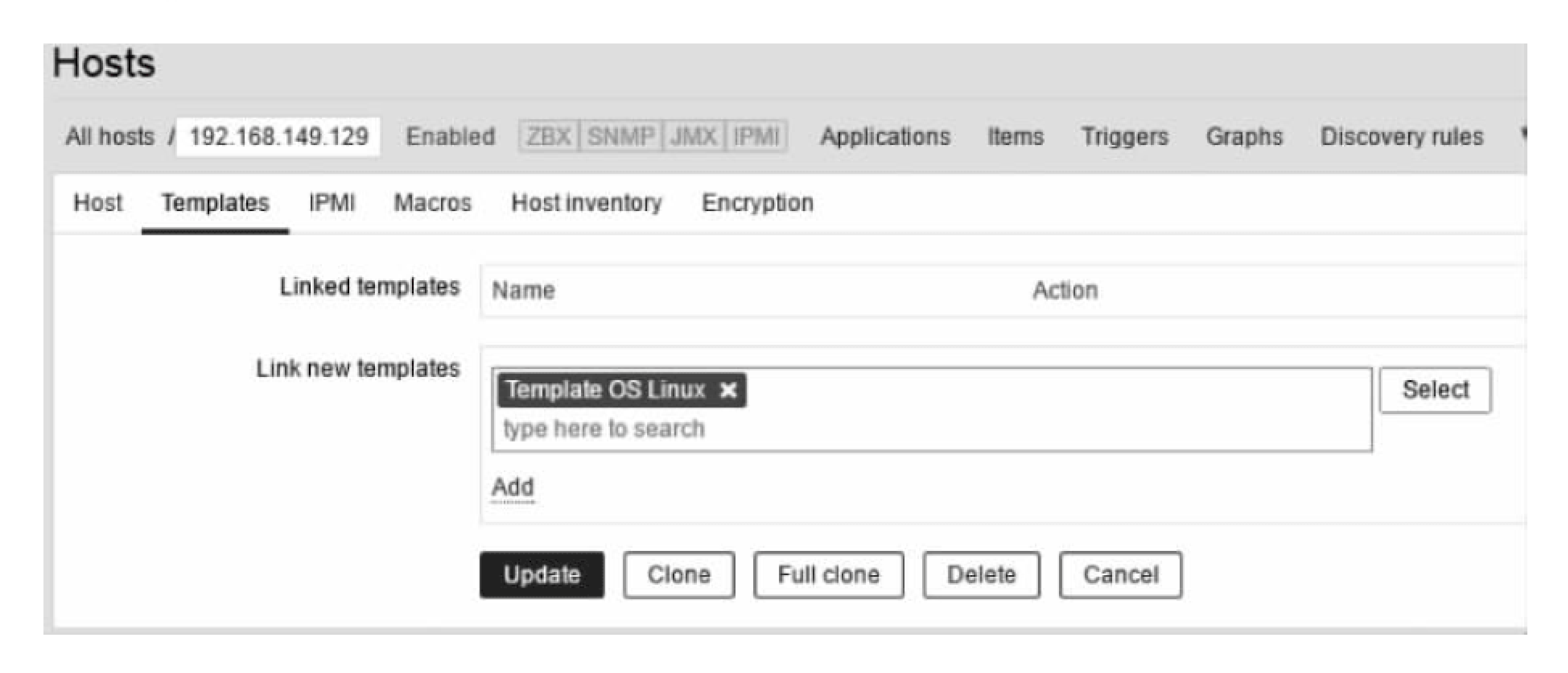
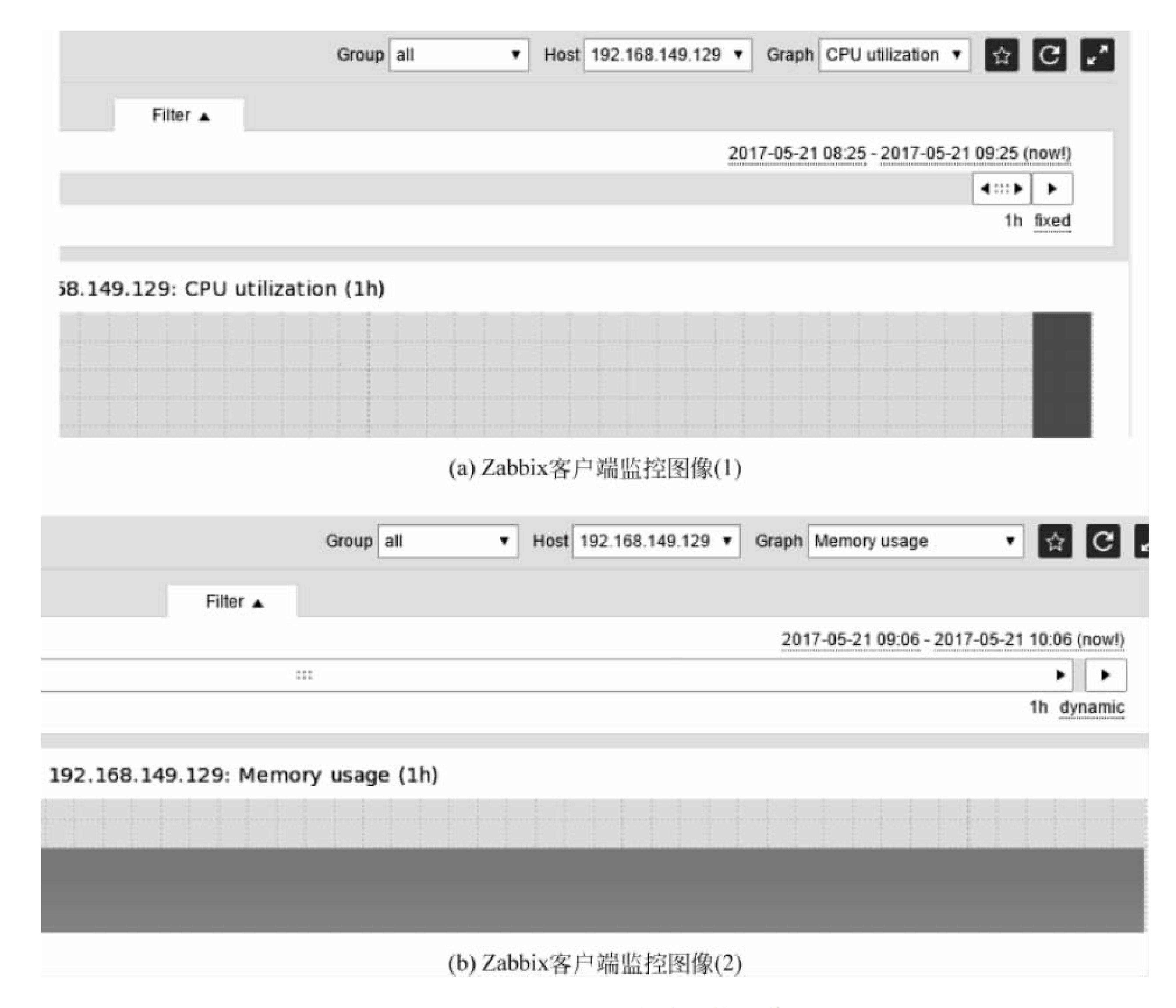


图 13-12　Zabbix 为客户端监控添加模板

依次选择 Zabbix Web→Monitoring→Graphs→Group→Host→Graph，监控图像如图 13-13 所示。

如果无法监控到客户端，可以在 Zabbix server 端，执行命令获取 agent 的 items key 值是否有返回，例如 system. uname 为返回客户端的 uname 信息，命令如下：

```
/usr/local/zabbix/bin/zabbix_get  -s 192.168.149.130  -k system.uname
```



(a) Zabbix客户端监控图像(1)

(b) Zabbix客户端监控图像(2)

图 13-13　Zabbix 客户端监控图像

13.6　Zabbix 配置文件详解

Zabbix 监控系统组件分为 server、proxy、agentd 端，对各自组件的参数进行详细了解，能够更加深入理解 Zabbix 监控功能以及对 Zabbix 进行调优，3 个组件的常用参数详解如下：

（1）zabbix_server.conf 配置文件参数详解如下：

- DBHost：数据库主机地址。
- DBName：数据库名称。
- DBPassword：数据库密码。
- DBPort：数据库端口，默认为 3306。
- AlertScriptsPath：告警脚本存放路径。
- CacheSize：存储监控数据的缓存。

- CacheUpdateFrequency：更新一次缓存时间。
- DebugLevel：日志级别。
- LogFile：日志文件。
- LogFileSize：日志文件大小，超过自动切割。
- LogSlowQueries：数据库慢查询记录，单位为 ms。
- PidFile：PID 文件。
- ProxyConfigFrequency：proxy 被动模式下，server 用多少秒同步配置文件至 proxy。
- ProxyDataFrequency：被动模式下，server 间隔多少秒向 proxy 请求历史数据。
- StartDiscoverers：发现规则线程数。
- Timeout：连接 agent 超时时间。
- TrendCacheSize：历史数据缓存大小。
- User：Zabbix 运行的用户。
- HistoryCacheSize：历史记录缓存大小。
- ListenIP：监听本机的 IP 地址。
- ListenPort：监听端口。
- LoadModule：模块名称。
- LoadModulePath：模块路径。

（2）zabbix_proxy.conf 配置文件参数详解如下：

- ProxyMode：proxy 工作模式，默认为主动模式，主动发送数据至 server。
- Server：指定 server 端地址。
- ServerPort：server 端 port。
- Hostname：proxy 端主机名。
- ListenPort：proxy 端监听端口。
- LogFile：proxy 代理端日志路径。
- PidFile：PID 文件的路径。
- DBHost：proxy 端数据库主机名。
- DBName：proxy 端数据库名称。
- DBUser：proxy 端数据库用户。
- DBPassword：proxy 端数据库密码。
- DBSocket：proxy 数据库 socket 路径。
- DBPort：proxy 数据库端口号。
- DataSenderFrequency：proxy 向 server 发送数据的时间间隔。
- StartPollers：proxy 程池数量。
- StartDiscoverers：proxy 端自动发现主机的线程数量。
- CacheSize：内存缓存配置。
- StartDBSyncers：同步数据线程数。

- HistoryCacheSize：历史数据缓存大小。
- LogSlowQueries：慢查询日志记录，单位为 ms。
- Timeout：超时时间。

(3) zabbix_agentd. conf 配置文件参数详解如下：

- EnableRemoteCommands：运行服务端远程至客户端执行命令或者脚本。
- Hostname：客户端主机名。
- ListenIP：监听的 IP 地址。
- ListenPort：客户端监听端口。
- LoadModulePath：模块路径。
- LogFile：日志文件路径。
- PidFile：PID 文件名。
- Server：指定 server IP 地址。
- ServerActive：Zabbix 主动监控 server 的 IP 地址。
- StartAgents：agent 启动进程，如果设置为 0，表示禁用被动监控。
- Timeout：超时时间。
- User：运行 Zabbix 的用户。
- UserParameter：用户自定义 key。
- BufferSize：缓冲区大小。
- DebugLevel：Zabbix 日志级别。

13.7 Zabbix 自动发现及注册

熟练掌握 Zabbix 监控平台监控单台客户端之后，假设企业中有成千上万台服务器，如果手工添加会非常耗时，造成大量人力成本的浪费，有没有更好的自动化添加客户端的方法呢？

Zabbix 自动发现功能是为了解决批量监控而设计的，那么什么是自动发现呢？简单来说，就是 Zabbix server 端可以基于设定的规则，自动批量的去发现局域网若干服务器，并自动把服务器添加至 Zabbix 监控平台，省去人工手动频繁的添加，节省大量的人力成本。

Zabbix 相对于 Nagios、Cacti 监控来说，如果想实现批量监控，Nagios、Cacti 需要手动单个添加设备、分组、项目、图像，可以使用脚本，但是不能实现自动发现方式添加。

Zabbix 最大的特点之一是可以批量自动发现主机并监控，利用发现(discovery)模块，实现自动发现主机、自动将主机添加到主机组、自动加载模板、自动创建项目(items)、自动创建监控图像，操作步骤如下：

(1) 依次选择 Configuration→Discovery → Create discovery rule，创建客户端发现规则，如图 13-14 所示。

具体参数说明如下：

Name：规则名称。

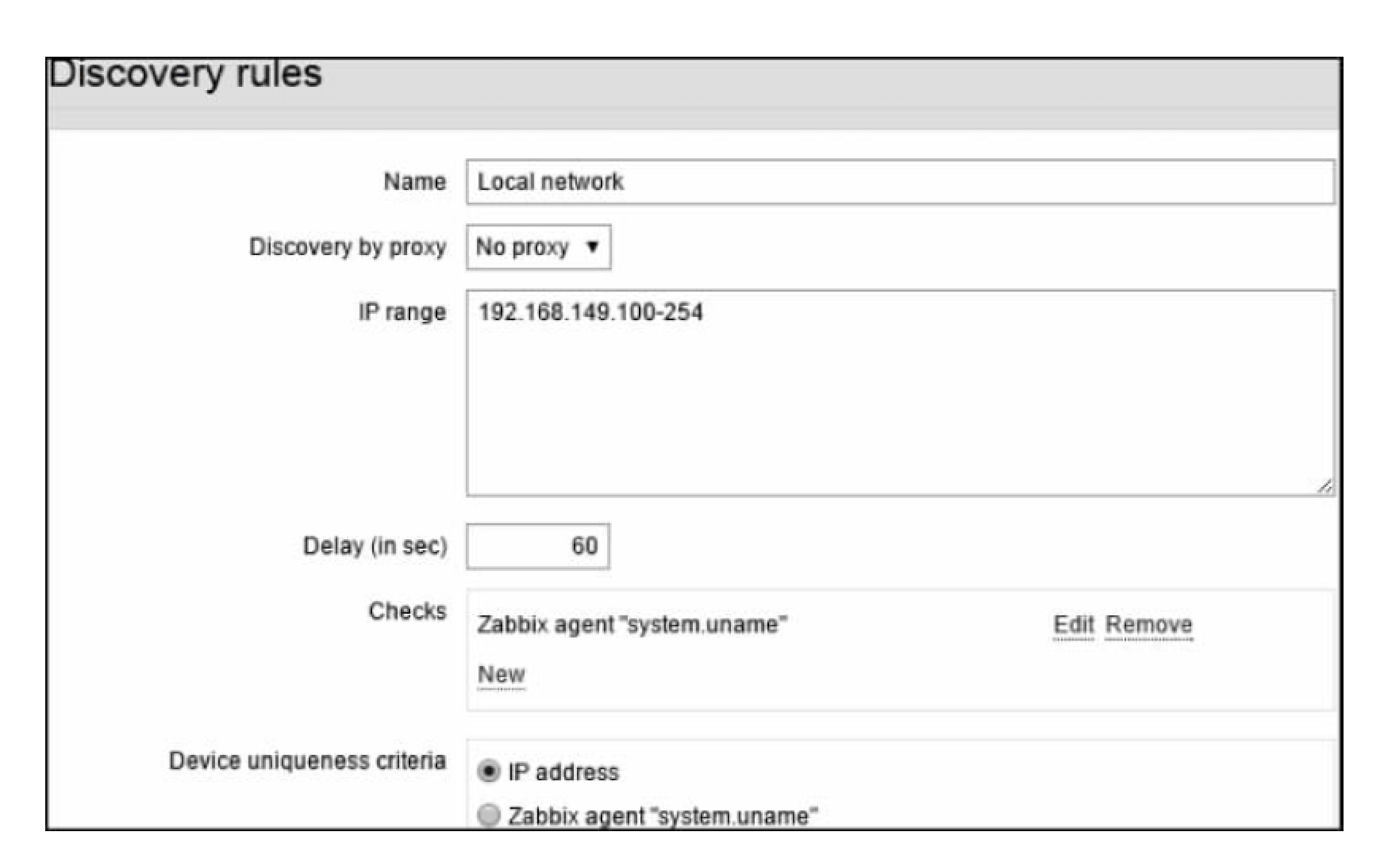


图 13-14　创建客户端发现规则

- Discovery by proxy：通过代理探索。
- IP range：zabbix_server 探索区域的 IP 范围。
- Delay：搜索一次的时间间隔。
- Checks：检测方式，如用 ping 方式去发现主机，zabbix_server 需安装 fping，此处使用 agent 方式发现。
- Device uniqueness criteria：以 IP 地址作为被发现主机的标识。

(2) Zabbix 客户端安装 agent。

由于发现规则里选择 checks 方式为 agent，所以需在所有被监控的服务器安装 Zabbix agent，安装的方法可以手动安装，也可以使用 shell 脚本，附 Zabbix 客户端安装脚本，脚本运行方法为 sh auto_install_zabbix.sh。

```
#!/bin/bash
#auto install zabbix
#by jfedu.net 2017
#############
ZABBIX_SOFT="zabbix-3.2.6.tar.gz"
INSTALL_DIR="/usr/local/zabbix/"
SERVER_IP="192.168.149.128"
IP='ifconfig|grep Bcast|awk '{print $2}'|sed 's/addr://g''
AGENT_INSTALL(){
yum -y install curl curl-devel net-snmp net-snmp-devel perl-DBI
groupadd zabbix ; useradd -g zabbix zabbix; usermod -s /sbin/nologin zabbix
tar -xzf $ZABBIX_SOFT; cd 'echo $ZABBIX_SOFT|sed 's/.tar.*//g''
./configure --prefix=/usr/local/zabbix --enable-agent&&make install
```

```
if [ $? -eq 0 ]; then
    ln -s /usr/local/zabbix/sbin/zabbix_* /usr/local/sbin/
fi
cd - ; cd zabbix-3.2.6
cp misc/init.d/tru64/zabbix_agentd /etc/init.d/zabbix_agentd ; chmod o+x /etc/init.d/zabbix_
agentd
#config zabbix agentd
cat >$INSTALL_DIR/etc/zabbix_agentd.conf << EOF
LogFile=/tmp/zabbix_agentd.log
Server= $SERVER_IP
ServerActive= $SERVER_IP
Hostname = $IP
EOF
#start zabbix agentd
/etc/init.d/zabbix_agentd restart
/etc/init.d/iptables stop
setenforce 0
}
AGENT_INSTALL
```

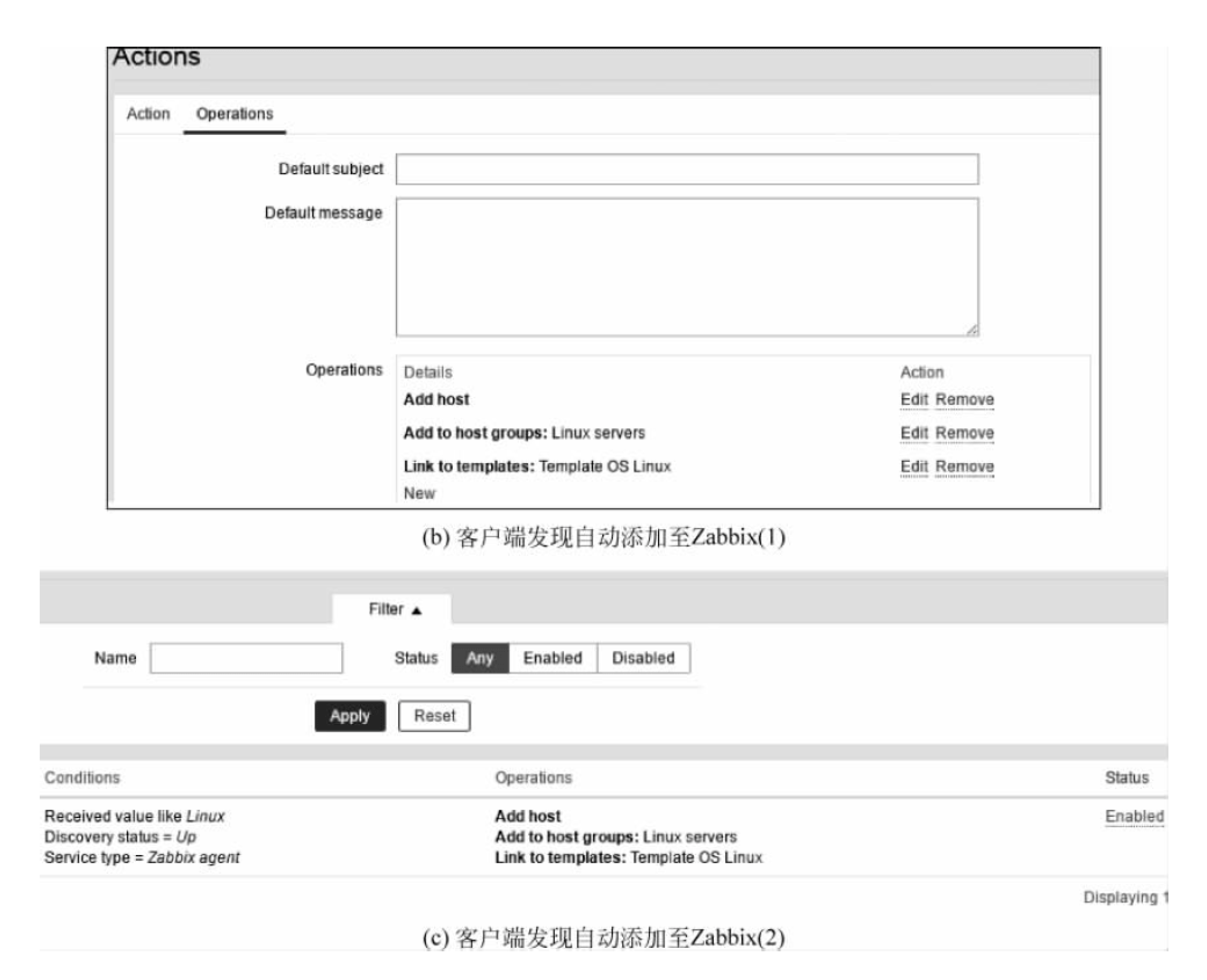
（3）创建发现 Action。

Zabbix 发现规则创建完毕，客户端 agent 安装完后，被发现的 IP 主机不会自动添加至 Zabbix 监控列表，需要添加发现动作，依次选择 Configuration→Actions→Event source (Discovery)→Create action 即可。

添加规则时，系统默认存在一条发现规则，可以新建规则，也可以编辑默认规则，如图 13-15 所示，编辑默认发现规则，单击 Operations 设置发现操作，分别设置 Add host、Add to host groups、Link to templates，最后启用规则即可。

(a) 创建客户端发现动作

图 13-15　设置发现操作

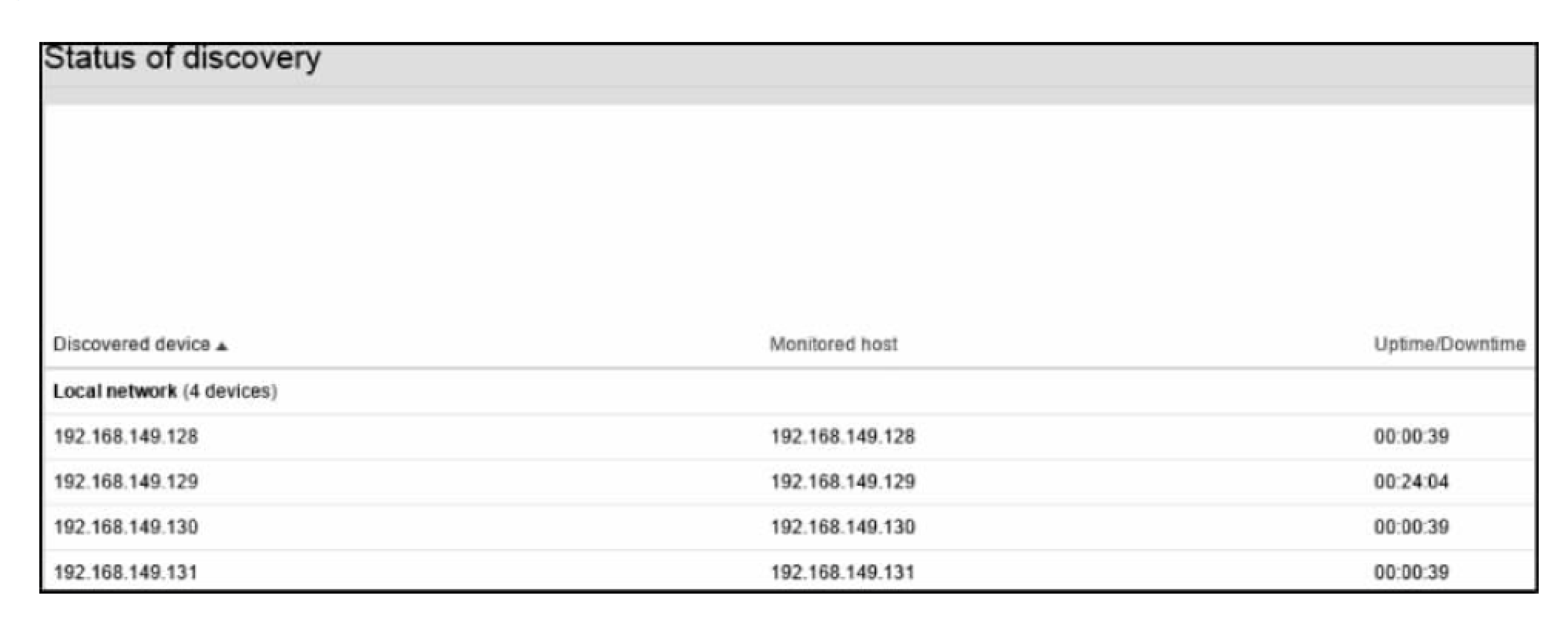


(b) 客户端发现自动添加至Zabbix(1)

(c) 客户端发现自动添加至Zabbix(2)

图 13-15 （续）

依次选择 Monitoring→Discovery，查看通过发现规则找到的服务器 IP 列表，如图 13-16 所示。



Status of discovery

Discovered device ▲	Monitored host	Uptime/Downtime
Local network (4 devices)		
192.168.149.128	192.168.149.128	00:00:39
192.168.149.129	192.168.149.129	00:24:04
192.168.149.130	192.168.149.130	00:00:39
192.168.149.131	192.168.149.131	00:00:39

图 13-16　被发现的客户端列表

依次选择 Configuration→Hosts，查看 4 台主机是否被自动监控至 Zabbix 监控平台，如图 13-17 所示。

ZABBIX　Monitoring　Inventory　Reports　Configuration　Administration

Host groups　Templates　Hosts　Maintenance　Actions　Event correlation　Discovery　IT services

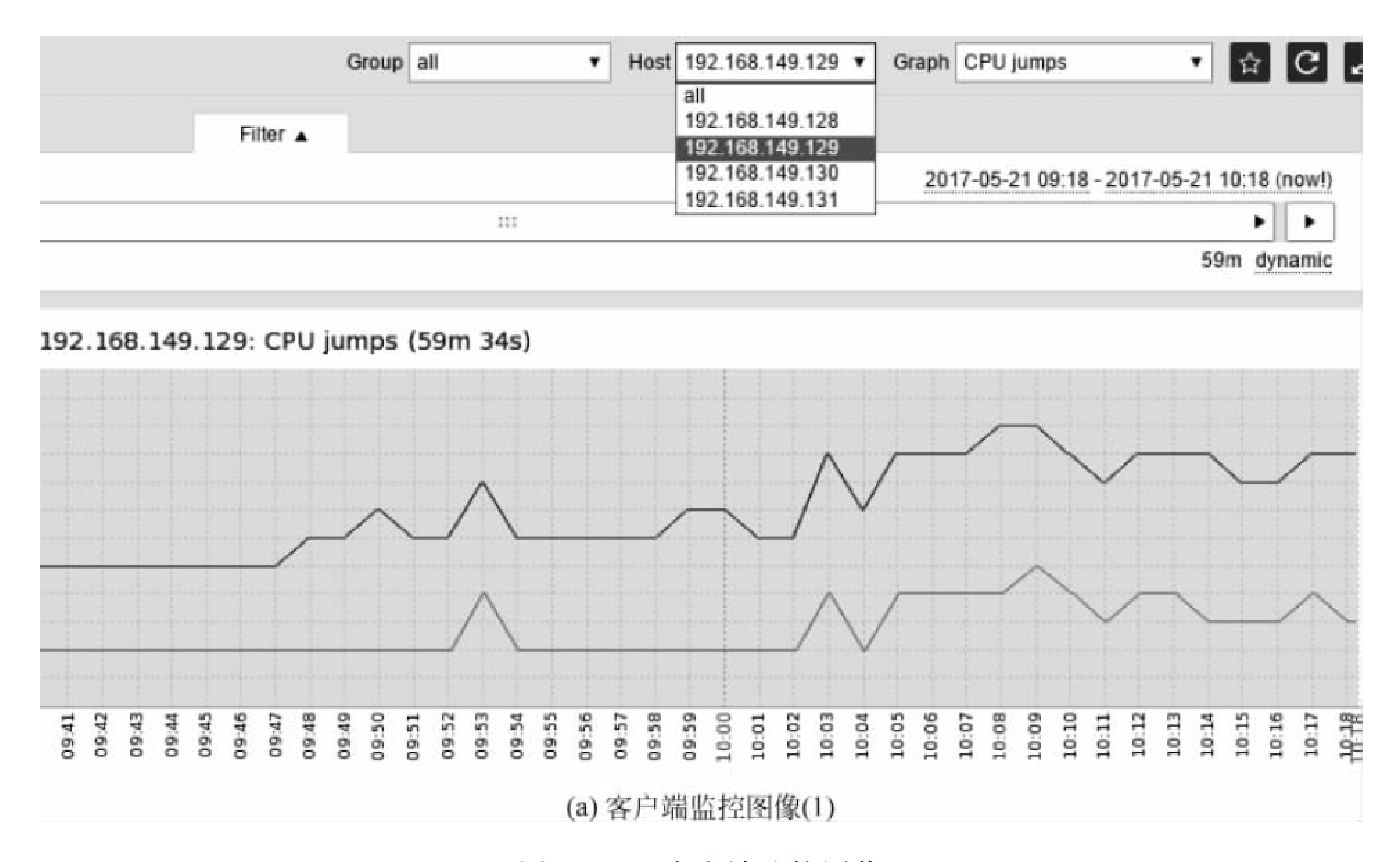
Hosts

Filter ▼

Name ▲	Applications	Items	Triggers	Graphs	Discovery	Web	Interface	Templates
192.168.149.128	Applications 10	Items 32	Triggers 15	Graphs 5	Discovery 2	Web	192.168.149.128: 10050	Template OS
192.168.149.129	Applications 10	Items 44	Triggers 19	Graphs 8	Discovery 2	Web	192.168.149.129: 10050	Template OS
192.168.149.130	Applications 10	Items 32	Triggers 15	Graphs 5	Discovery 2	Web	192.168.149.130: 10050	Template OS
192.168.149.131	Applications 10	Items 32	Triggers 15	Graphs 5	Discovery 2	Web	192.168.149.131: 10050	Template OS

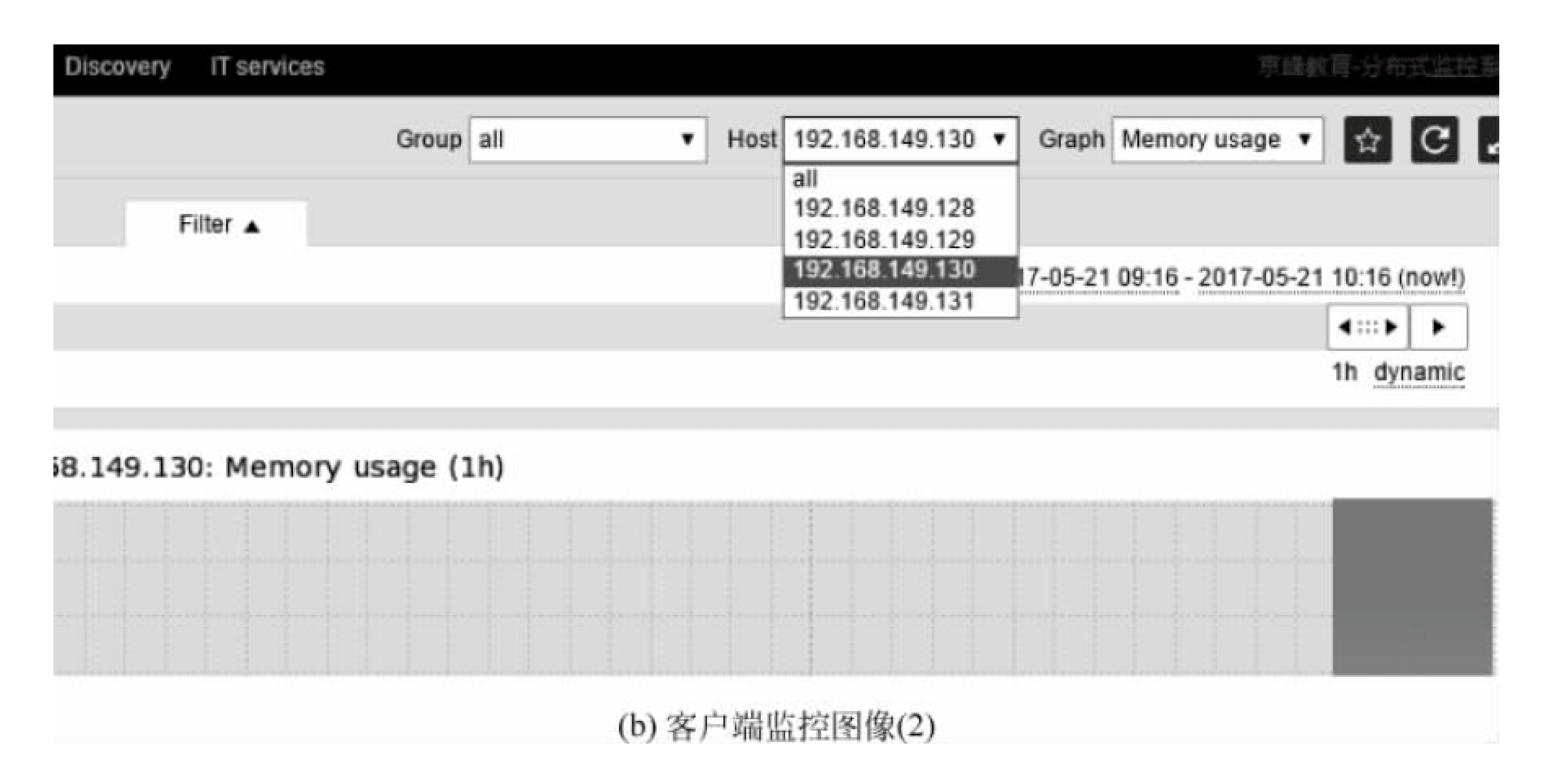
图 13-17　自动发现的主机被添加至 Hosts 列表

依次选择 Monitoring→Graphs，监控图像查看，如图 13-18 所示，可以选择 Host、Graph 分别查看各种监控图像。



(a) 客户端监控图像(1)

图 13-18　客户端监控图像



(b) 客户端监控图像(2)

图 13-18 （续）

13.8 Zabbix 邮件报警

Zabbix 服务端、客户端都已经部署完成，被监控主机已经添加，Zabbix 监控运行正常，查看 Zabbix 监控服务器，可以了解服务器的运行状态是否正常，运维人员不会时刻登录 Zabbix 监控平台查看服务器的状态。

可以在 Zabbix 服务端设置邮件报警，当被监控主机宕机或者达到设定的触发器预设值时，会自动发送报警邮件、微信信息到指定的人员，有利于运维人员收到信息后第一时间解决故障。Zabbix 邮件报警设置步骤如下：

（1）设置邮件模板及邮件服务器。

依次选择 Administration→Media types→Create media type，填写邮件服务器信息，根据提示设置完毕，如图 13-19 所示。

（2）配置接收报警的邮箱。

依次选择 Administration→user→Admin(Zabbix Administrator)→user→admin，再选择 Media，单击 Add 添加发送邮件的类型为 Email，同时指定接收邮箱地址为 wgkgood@163.com，根据实际需求可以改成自己的接收人，如图 13-20 所示。

（3）添加报警触发器。

依次选择 Configuration→Actions→Action→Event source→Triggers→Create Action，如图 13-21 所示，分别设置 Action、Operations、Recovery operations。具体参数设置如下：

- 选择 Action→New condition，勾选 Trigger severity>=Warning；
- 在 Operations 中设置报警间隔为 60s，自定义报警信息，报警信息发送至 administrators 组；



图 13-19 Zabbix 邮件报警邮箱设置

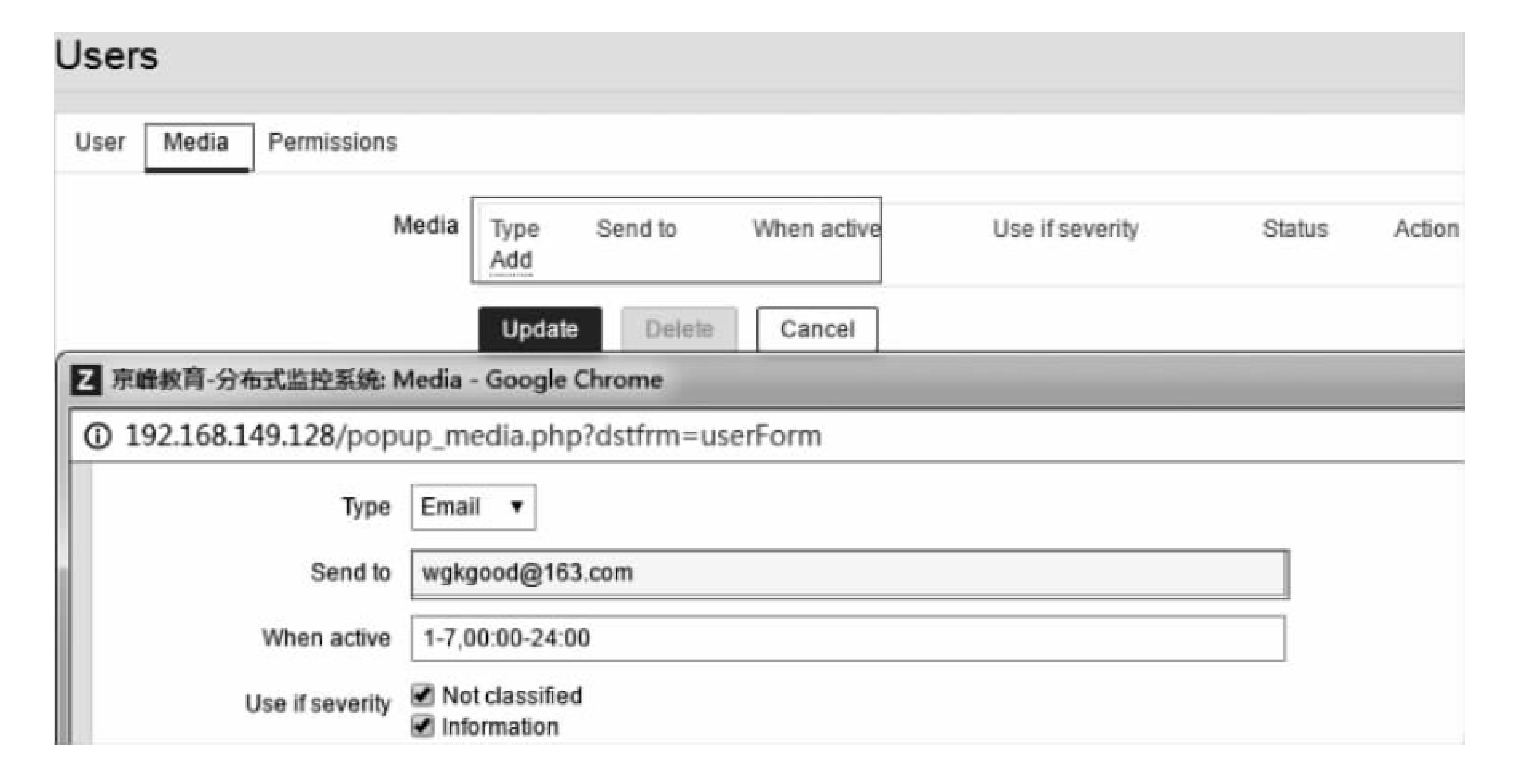


图 13-20 Zabbix 邮件报警添加接收人

Actions

Action　Operations　Recovery operations

Name　Report problems to Zabbix administrators

Conditions　Label　Name　Action

New condition　Trigger severity　>=　Warning

Add

Enabled

Update　Clone　Delete　Cancel

(a) 邮件报警Action设置

Action　Operations　Recovery operations

Default operation step duration　60　(minimum 60 seconds)

Default subject　故障{TRIGGER.STATUS},服务器:{HOSTNAME1}发生: {TRIGGER.NAME}故障!

Default message
告警主机:{HOSTNAME1}
告警时间:{EVENT.DATE} {EVENT.TIME}
告警等级:{TRIGGER.SEVERITY}
告警信息: {TRIGGER.NAME}
告警项目:{TRIGGER.KEY1}
问题详情:{ITEM.NAME}:{ITEM.VALUE}

Pause operations while in maintenance

Operations　Steps　Details　Start in
1　**Send message to user groups:** Zabbix administrators via all media　Immedi
New

(b) 邮件报警Operations设置

Actions

Action　Operations　Recovery operations

Default subject　恢复{TRIGGER.STATUS}, 服务器:{HOSTNAME1}: {TRIGGER.NAME}已恢复!

Default message
告警主机:{HOSTNAME1}
告警时间:{EVENT.DATE} {EVENT.TIME}
告警等级:{TRIGGER.SEVERITY}
告警信息: {TRIGGER.NAME}
告警项目:{TRIGGER.KEY1}
问题详情:{ITEM.NAME}:{ITEM.VALUE}

Operations　Details　A
Notify all who received any messages regarding the problem before　E
New

Update　Clone　Delete　Cancel

(c) 邮件报警Recovery operations设置

图 13-21　邮件报警设置

- 在 Recovery operations 中自定义恢复信息，恢复信息发送至 administrators 组。

报警邮件标题可以使用默认信息，也可使用如下中文报警内容。

```
名称：Action－Email
默认标题：故障{TRIGGER.STATUS},服务器:{HOSTNAME1}发生：{TRIGGER.NAME}故障!
默认信息：
告警主机:{HOSTNAME1}
告警时间:{EVENT.DATE} {EVENT.TIME}
告警等级:{TRIGGER.SEVERITY}
告警信息：{TRIGGER.NAME}
告警项目:{TRIGGER.KEY1}
问题详情:{ITEM.NAME}:{ITEM.VALUE}
当前状态:{TRIGGER.STATUS}:{ITEM.VALUE1}
事件 ID:{EVENT.ID}
```

恢复邮件标题可以使用默认信息，也可使用如下中文报警恢复内容。

```
恢复标题：恢复{TRIGGER.STATUS}, 服务器:{HOSTNAME1}：{TRIGGER.NAME}已恢复!
恢复信息：
告警主机:{HOSTNAME1}
告警时间:{EVENT.DATE} {EVENT.TIME}
告警等级:{TRIGGER.SEVERITY}
告警信息：{TRIGGER.NAME}
告警项目:{TRIGGER.KEY1}
问题详情:{ITEM.NAME}:{ITEM.VALUE}
当前状态:{TRIGGER.STATUS}:{ITEM.VALUE1}
事件 ID:{EVENT.ID}
```

依次选择 Monitoring→Problems，检查有问题的 Action 事件，单击 Time 下方时间，如图 13-22 所示，可以查看邮件执行成功还是失败。



(a) Zabbix查看有问题的事件

图 13-22　检查有问题的 Action 事件



(b) Zabbix有问题的事件执行任务

图 13-22　(续)

Zabbix 邮件发送失败，报错 Support for SMTP authentication was not compiled in，原因是 Zabbix CURL 版本要求至少是 7.20＋版本，因此需升级 CURL，升级方法如下：

```
wget http://mirror.city-fan.org/ftp/contrib/yum-repo/city-fan.org-release-1-13.rhel6.
noarch.rpm
rpm -ivh city-fan.org-release-1-13.rhel6.noarch.rpm
yum upgrade libcurl -y
curl -V
```

CURL 升级完毕之后，测试邮件发送，还是报同样的错误，原因是需要重新将 Zabbix server 服务通过源码编译安装一遍，安装完 Zabbix server 并重启服务，有可能会出现乱码，乱码问题是由于数据库字符集需改成 UTF-8 格式，将 Zabbix 数据库导出，然后修改字符集 latin1 为 utf8，再将 SQL 导入，重启 Zabbix 即可，最终如图 13-23 所示。

Time	Type	Status	Retries left	Recipient	Message
m					
05/21/2017 12:15:03 PM	Email	Sent		Admin (Zabbix Administrator) wgkgood@163.com	故障PROBLEM,服务器:192.168.149.130发生: Free disk spa than 20% on volume /boot故障! 告警主机:192.168.149.130 告警时间:2017.05.21 12:15:01 告警等级:Warning 告警信息: Free disk space is less than 20% on volume /boot 告警项目:vfs.fs.size[/boot,pfree] 问题详情:Free disk space on /boot (percentage):0 % 当前状态:PROBLEM:0 % 事件ID:226

(a) Zabbix事件发送邮件进程

图 13-23　Zabbix 监控发送邮件

故障PROBLEM,服务器:192.168.149.130发生: Free disk space is less than 20% on volume /boot故障!
发件人：wgk<wgk@jfedu.net>
收件人：我<wgkgood@163.com>
时 间：2017年05月21日 12:15 (星期日)

告警主机:192.168.149.130
告警时间:2017.05.21 12:15:01
告警等级:Warning
告警信息: Free disk space is less than 20% on volume /boot
告警项目:vfs.fs.size[/boot,pfree]
问题详情:Free disk space on /boot (percentage):0 %
当前状态:PROBLEM:0 %
事件ID:226

(b) Zabbix监控故障item发送报警邮件

恢复OK, 服务器:192.168.149.130: Free disk space is less than 20% on volume /boot已恢复!
发件人：wgk<wgk@jfedu.net>
收件人：我<wgkgood@163.com>
时 间：2017年05月21日 12:13 (星期日)

告警主机:192.168.149.130
告警时间:2017.05.21 12:03:01
告警等级:Warning
告警信息: Free disk space is less than 20% on volume /boot
告警项目:vfs.fs.size[/boot,pfree]
问题详情:Free disk space on /boot (percentage):85.74 %
当前状态:OK:85.74 %
事件ID:194

(c) Zabbix监控故障item恢复发送邮件

图 13-23 （续）

13.9 Zabbix 监控 MySQL 主从复制

Zabbix 监控除了可以使用 agent 监控客户端服务器状态、CPU、内存、硬盘、网卡流量等运行情况，同时 Zabbix 还可以监控 MySQL 主从复制、LAMP、Nginx Web 服务器等，以下为 Zabbix 监控 MySQL 主从复制的步骤：

（1）在 Zabbix agent 端/data/sh 目录创建 shell 脚本 mysql_ab_check.sh，写入如下代码：

```
#!/bin/bash
/usr/local/mysql/bin/mysql -uroot -e 'show slave status\G' |grep -E
"Slave_IO_Running|Slave_SQL_Running"|awk '{print $2}'|grep -c Yes
```

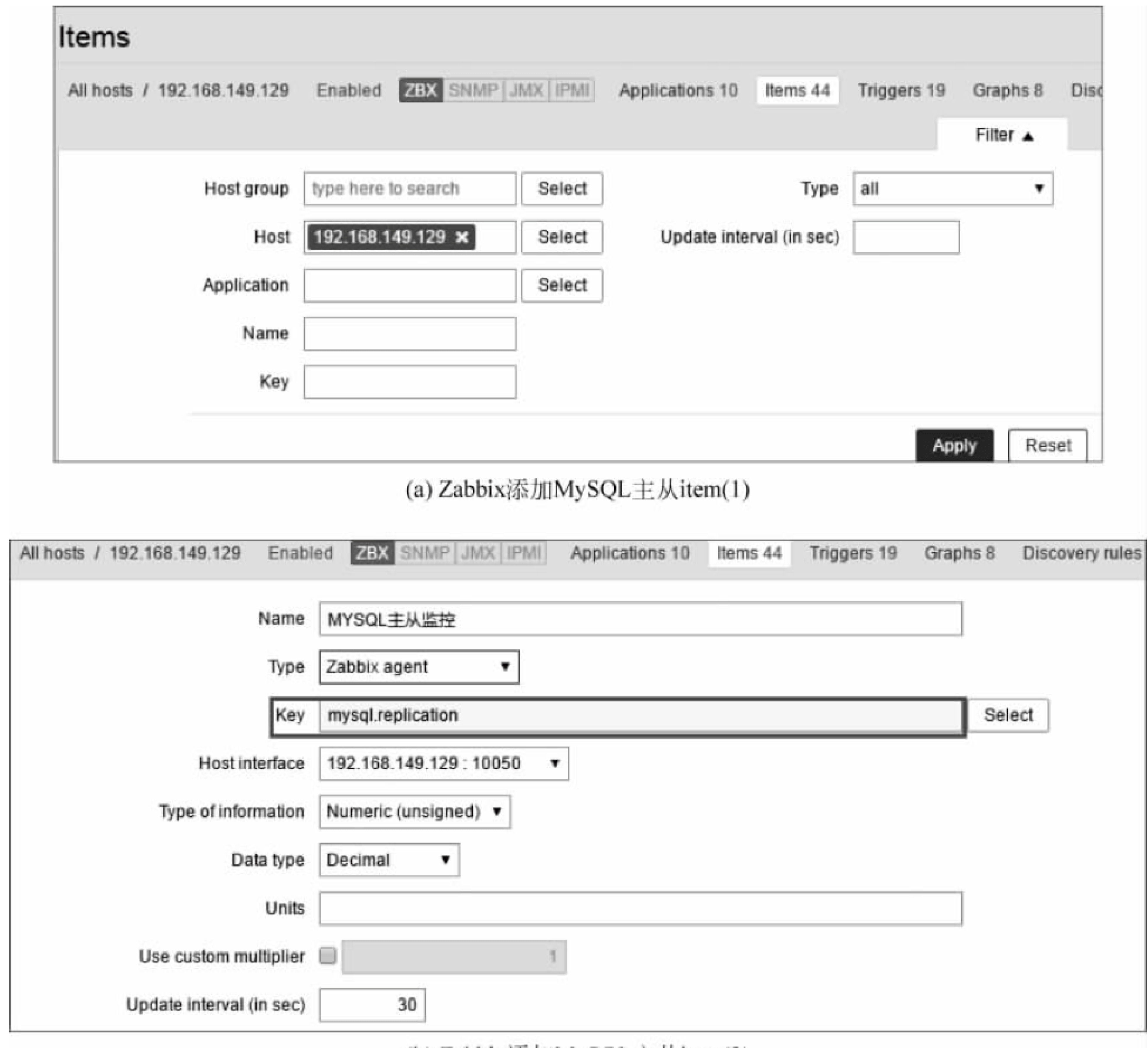
(2) 在客户端 zabbix_agentd.conf 配置文件中加入如下代码:

```
UserParameter = mysql.replication,sh /data/sh/mysql_ab_check.sh
```

(3) Zabbix 服务器端获取监控数据,如果返回值为 2,则证明从库 I/O、SQL 线程均为 Yes,表示主从同步成功,代码如下:

```
/usr/local/zabbix/bin/zabbix_get -s 192.168.149.129 -k mysql.replication
```

(4) Zabbix Web 平台,在 192.168.149.129 hosts 中创建 item 监控项,单击右上角 create item,在 Key 输入栏中填写 zabbix_agentd 配置文件中的 mysql.replication 即可,如图 13-24 所示。



(a) Zabbix添加MySQL主从item(1)

(b) Zabbix添加MySQL主从item(2)

图 13-24　Zabbix 添加 MySQL 主从 item

MySQL 主从监控项创建 Graph 图像，如图 13-25 所示。

(a) 创建MySQL主从监控图像(1)

(b) 创建MySQL主从监控图像(2)

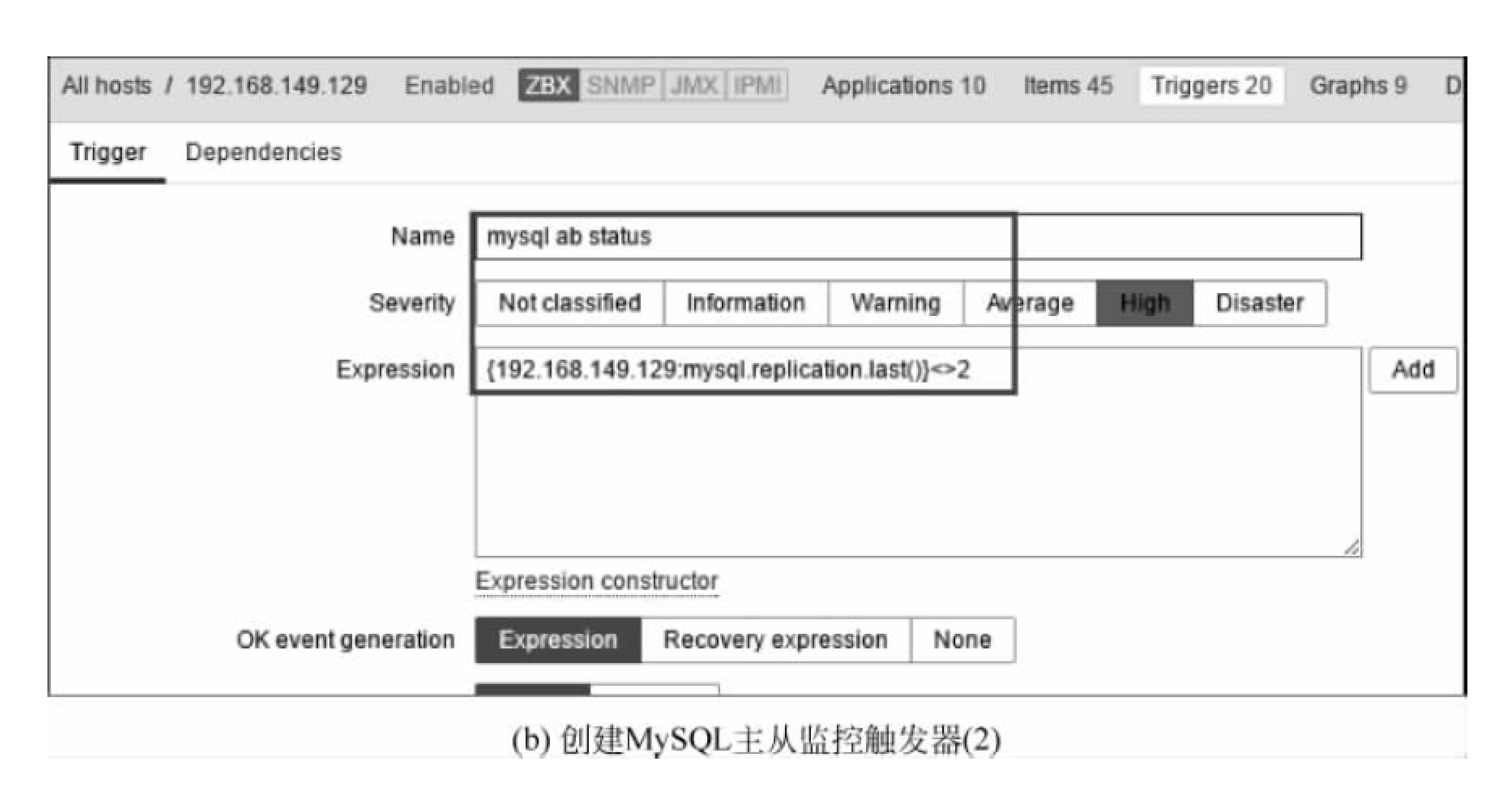
图 13-25　创建 MySQL 主从监控图像

MySQL 主从监控项创建触发器，如图 13-26 所示，MySQL 主从状态监控，设置触发器条件 Key 值不等于 2 即可，不等于 2 即表示 MySQL 主从同步状态异常，匹配触发器会执行 Actions。



(a) 创建MySQL主从监控触发器(1)

图 13-26　创建 MySQL 主从监控触发器



(b) 创建MySQL主从监控触发器(2)

图 13-26 （续）

如果主从同步状态异常，Key 值不等于 2，会触发邮件报警，报警信息如图 13-27 所示。



图 13-27 MySQL 主从监控报警邮件

13.10 Zabbix 日常问题汇总

Zabbix 可以设置中文汉化，如果访问 Zabbix 出现如下历史记录乱码，即 Web 界面乱码，原因是数据库导入前不是 UTF-8 字符集，需要修改为 UTF-8 模式，如图 13-28 所示。

MySQL 数据库修改字符集方法，在 vim /etc/my.cnf 配置段加入如下代码：

```
[mysqld]
character-set-server= utf8
```

```
[client]
default-character-set = utf8
[mysql]
default-character-set = utf8
```

```
ype 'help;' or '\h' for help. Type '\c' to clear the current input statement.
ysql> show variables like "%char%";
+--------------------------+----------------------------+
| variable_name            | value                      |
+--------------------------+----------------------------+
| character_set_client     | latin1                     |
| character_set_connection | latin1                     |
| character_set_database   | latin1                     |
| character_set_filesystem | binary                     |
| character_set_results    | latin1                     |
| character_set_server     | latin1                     |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.04 sec)

ysql> 
```

图 13-28　数据库原字符集 latin1

备份 Zabbix 数据库并删除原数据库，重新创建，再导入备份的数据库，修改导入的 zabbix.sql 文件里面的 latin1 为 utf8，然后再导入到 Zabbix 数据库，乱码问题即可解决，代码如下：

```
sed -i 's/latin1/utf8/g' zabbix.sql
```

如果在查看 Graph 监控图像界面的时候出现乱码，如图 13-29 所示。

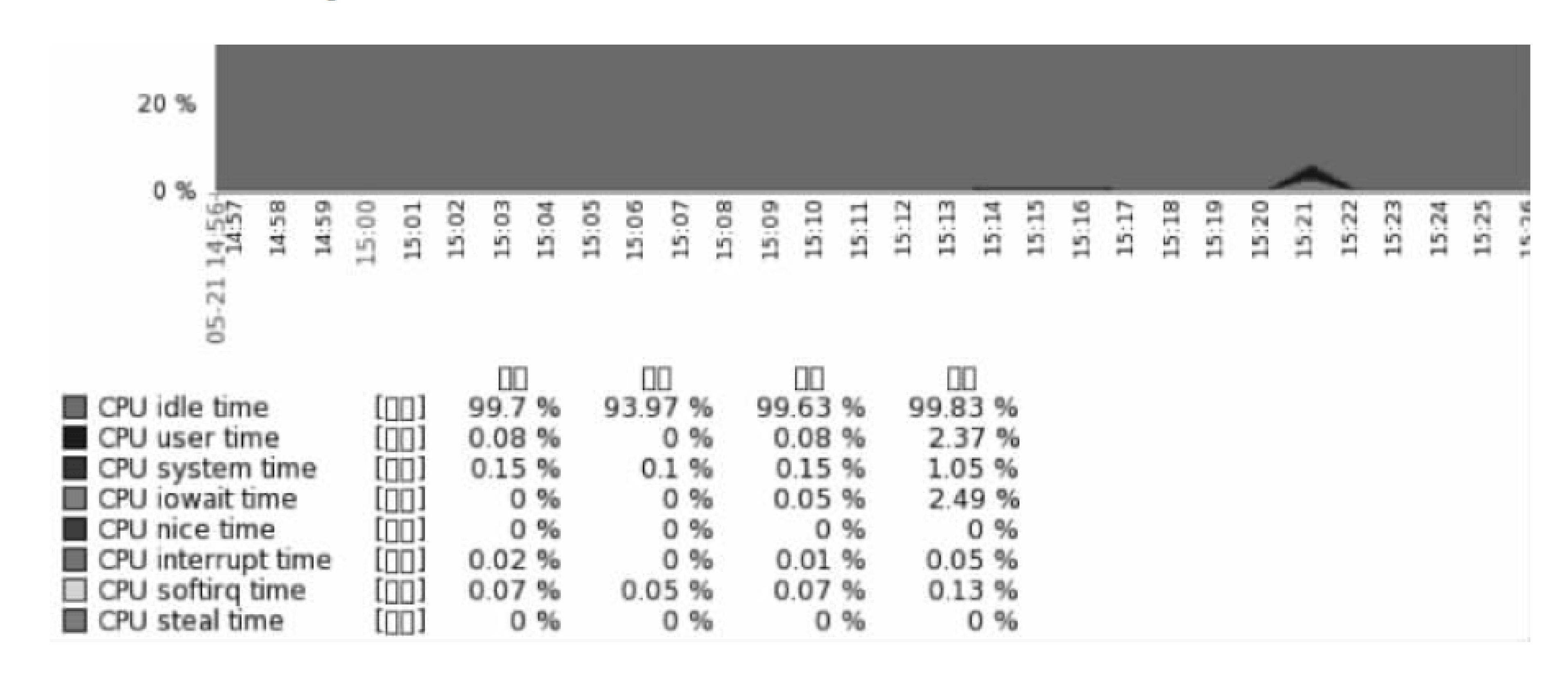


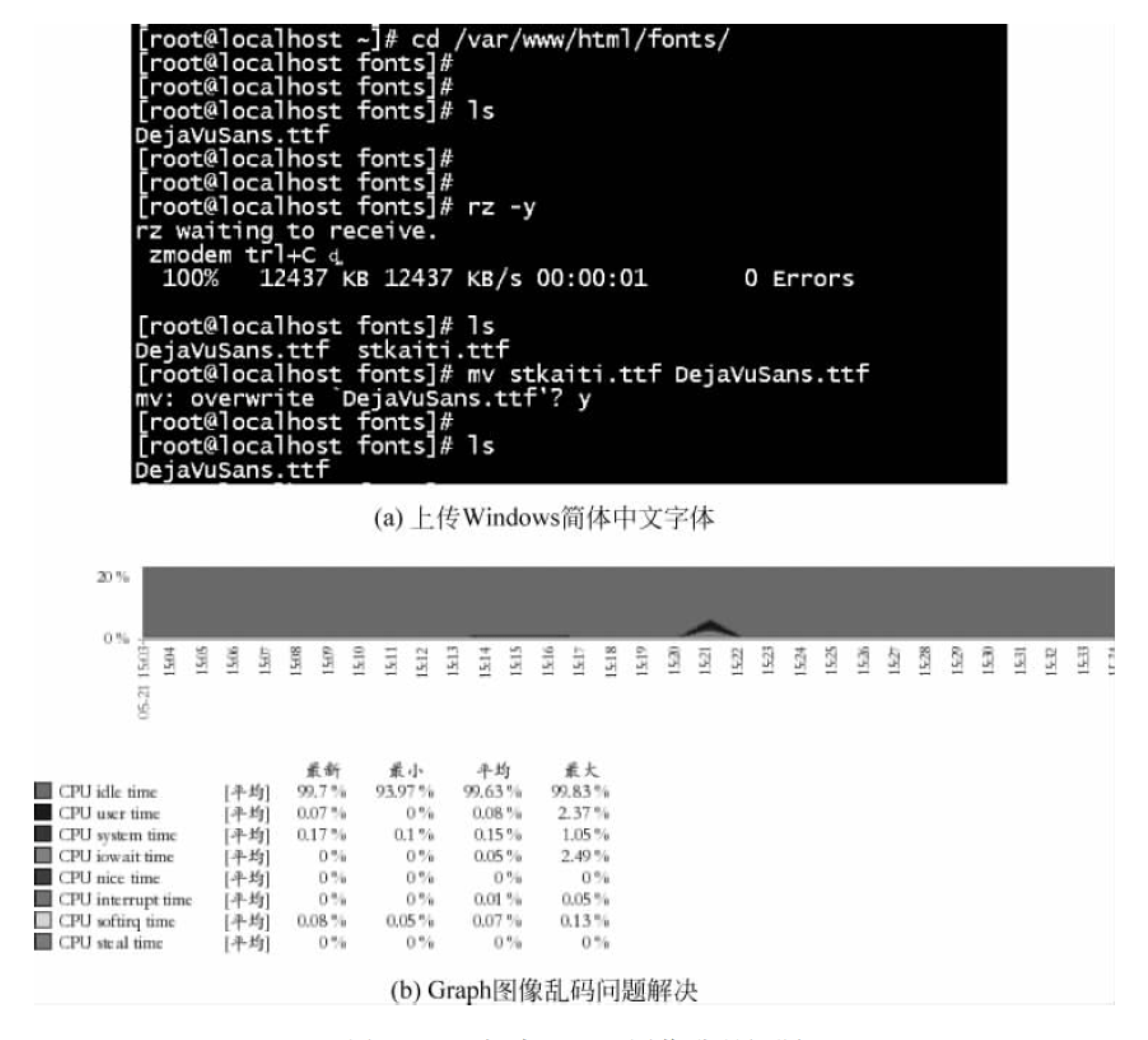
图 13-29　Graph 图像乱码

在 Windows 的控制面板中的“字体”选项中选择一种中文字库，例如“楷体”，如图 13-30 所示。

将字体文件 cp 至 Zabbix 服务 dauntfonts 目录下，即/var/www/html/zabbix/fonts，并且将 STKAITI.TTF 重命名为 DejaVuSans.ttf，刷新 Graph 图像，乱码问题即可解决，如图 13-31 所示。



图 13-30　选择 Windows 简体中文字体



(a) 上传Windows简体中文字体

(b) Graph图像乱码问题解决

图 13-31　解决 Graph 图像乱码问题

13.11 Zabbix 触发命令及脚本

Zabbix 在对服务或者设备进行监控的时候，如果被监控客户端服务异常，满足触发器，可以发送邮件报警、短信报警及微信报警。Zabbix 还可以远程执行命令或者脚本，对部分故障实现自动修复。具体可以执行的任务如下：

- 重启应用程序，例如 Apache、Nginx、MySQL、Tomcat 服务等；
- 通过 IPMI 接口重启服务器；
- 删除服务器磁盘空间及数据；
- 执行脚本及资源调度管理；
- 远程命令最大长度为 255 个字符；
- 同时支持多个远程命令；
- Zabbix 代理不支持远程命令。

使用 Zabbix 远程执行命令，需在 Zabbix 客户端配置文件开启对远程命令的支持，在 zabbix_agentd.conf 行尾加入如下代码，并重启服务，如图 13-32 所示。

```
EnableRemoteCommands = 1
```

```
[root@localhost ~]# cd /usr/local/zabbix/etc/
[root@localhost etc]#
[root@localhost etc]# ls
zabbix_agentd.conf  zabbix_agentd.conf.d
[root@localhost etc]#
[root@localhost etc]# vim zabbix_agentd.conf
LogFile=/tmp/zabbix_agentd.log
Server=192.168.149.128
ServerActive=192.168.149.128
Hostname = 192.168.149.129
UserParameter=mysql.replication,sh /data/sh/mysql_ab_check.sh
EnableRemoteCommands = 1
```

图 13-32　客户端配置远程命令支持

创建 Action，依次选择 Configuration→Actions→Triggers，如图 13-33 所示，类型选择 Remote command，Steps 表示执行命令 1～3 次，Step duration 设置每次命令执行间隔时间，60s 执行一次，执行命令方式选择 Zabbix agent，基于 sudo 执行命令即可。

在 Zabbix 客户端 sudoer 配置文件中添加 Zabbix 用户拥有执行权限且无须密码登录，代码如下：

```
Defaults:zabbix !requiretty
zabbix ALL = (ALL) NOPASSWD: ALL
```

在 Zabbix 客户端/data/sh/，创建 auto_clean_disk.sh，脚本代码如下：

```
#!/bin/bash
```



(a) 客户端触发器满足条件

(b) Operations类型选择Remote Command

图 13-33 创建 Action

```
#auto clean disk space
#2017 年 6 月 21 日 10:12:18
#by author jfedu.net
rm -rf /boot/test.img
find /boot/ -name "*.log" -size +100M -exec rm -rf {} \;
```

将 192.168.149.129 服务器/boot 目录临时写满，然后满足触发器，实现远程命令执行，查看问题事件命令执行结果，如图 13-34 所示。

如果 Zabbix 客户端脚本或者命令没有执行成功，HTTP 服务没有停止，可以在 Zabbix server 端执行如下命令，详情如图 13-35 所示。

```
/usr/local/zabbix/bin/zabbix_get -s 192.168.149.129 -k "system.run[sudo /etc/init.d/httpd restart]"
```



(a) Remote command执行成功

(b) Remote command执行磁盘清理成功

图 13-34　查看问题事件命令执行结果

```
ifconfig eth0
:Ethernet  HWaddr 00:0C:29:3F:F8:29
192.168.149.128  Bcast:192.168.149.255  Mask:255.255.255.0
: fe80::20c:29ff:fe3f:f829/64 Scope:Link
AST RUNNING MULTICAST  MTU:1500  Metric:1
:453673 errors:0 dropped:0 overruns:0 frame:0
:538109 errors:0 dropped:0 overruns:0 carrier:0
:0 txqueuelen:1000
168319859 (160.5 MiB)  TX bytes:239519442 (228.4 MiB)

/usr/local/zabbix/bin/zabbix_get -s 192.168.149.129 -k "system.run[sud
OK  ]
d: Could not reliably determine the server's fully qualified domain na
```

图 13-35　测试 Remote command 命令

13.12　Zabbix 分布式配置

Zabbix 是一个分布式监控系统，它可以以一个中心点、多个分节点的模式运行，使用 proxy 能大大地降低 Zabbix server 的压力，Zabbix proxy 可以运行在独立的服务器上，如图 13-36 所示。

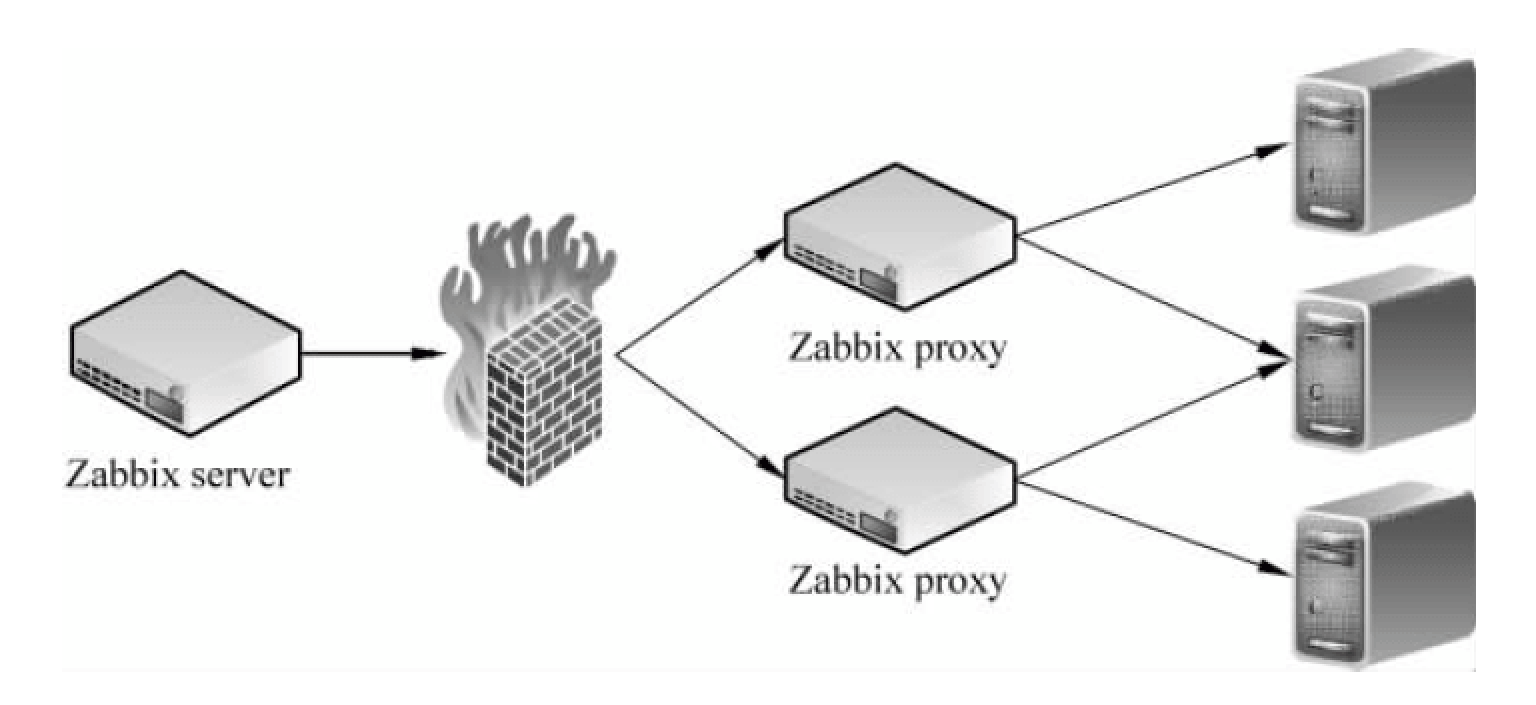


图 13-36 Zabbix proxy 网络拓扑图

安装 Zabbix proxy，基于 Zabbix-3.2.6.tar.gz 软件包，同时需要导入 Zabbix 基础框架表，具体实现方法如下：

(1) 下载 Zabbix 软件包，代码如下：

```
wget http://sourceforge.net/projects/zabbix/files/ZABBIX%20Latest%20Stable/3.2.6/zabbix-3.2.6.tar.gz/download
```

(2) 在 Zabbix proxy 上执行如下代码：

```
yum -y install curl curl-devel net-snmp net-snmp-devel perl-DBI
groupadd zabbix ; useradd -g zabbix zabbix; usermod -s /sbin/nologin zabbix
```

(3) Zabbix proxy 端配置。

创建 Zabbix 数据库，执行授权命令如下：

```
create database zabbix_proxy charset=utf8;
grant all on zabbix_proxy.* to zabbix@localhost identified by '123456';
flush privileges;
```

解压 Zabbix 软件包并将 Zabbix 基础 SQL 文件导入数据至 Zabbix 数据库，代码如下：

```
tar zxvf zabbix-3.2.6.tar.gz
cd zabbix-3.2.6
mysql -uzabbix -p123456 zabbix_proxy <database/mysql/schema.sql
mysql -uzabbix -p123456 zabbix_proxy <database/mysql/images.sql
```

切换至 Zabbix 解压目录，执行如下代码，安装 zabbix_server：

```
./configure --prefix=/usr/local/zabbix/ --enable-proxy --enable-agent --with-mysql
--enable-ipv6 --with-net-snmp --with-libcurl
make
make install
ln -s /usr/local/zabbix/sbin/zabbix_* /usr/local/sbin/
```

Zabbix proxy 安装完毕，cd /usr/local/zabbix/etc/目录，如图 13-37 所示。

```
[root@www-jfedu-net-129 ~]#
[root@www-jfedu-net-129 ~]# cd /usr/local/zabbix/
[root@www-jfedu-net-129 zabbix]# ls
bin  etc  lib  sbin  share
[root@www-jfedu-net-129 zabbix]# cd etc/
[root@www-jfedu-net-129 etc]# ls
zabbix_agentd.conf  zabbix_agentd.conf.d  zabbix_proxy.conf  zabbix_
[root@www-jfedu-net-129 etc]# ll
total 24
-rw-r--r-- 1 root root 10234 May 24 00:47 zabbix_agentd.conf
drwxr-xr-x 2 root root  4096 May 24 00:47 zabbix_agentd.conf.d
-rw-r--r-- 1 root root   220 May 24 00:52 zabbix_proxy.conf
drwxr-xr-x 2 root root  4096 May 24 00:48 zabbix_proxy.conf.d
[root@www-jfedu-net-129 etc]#
```

图 13-37　Zabbix proxy 安装目录

（4）备份 Zabbix proxy 配置文件，代码如下：

```
cp zabbix_proxy.conf zabbix_proxy.conf.bak
```

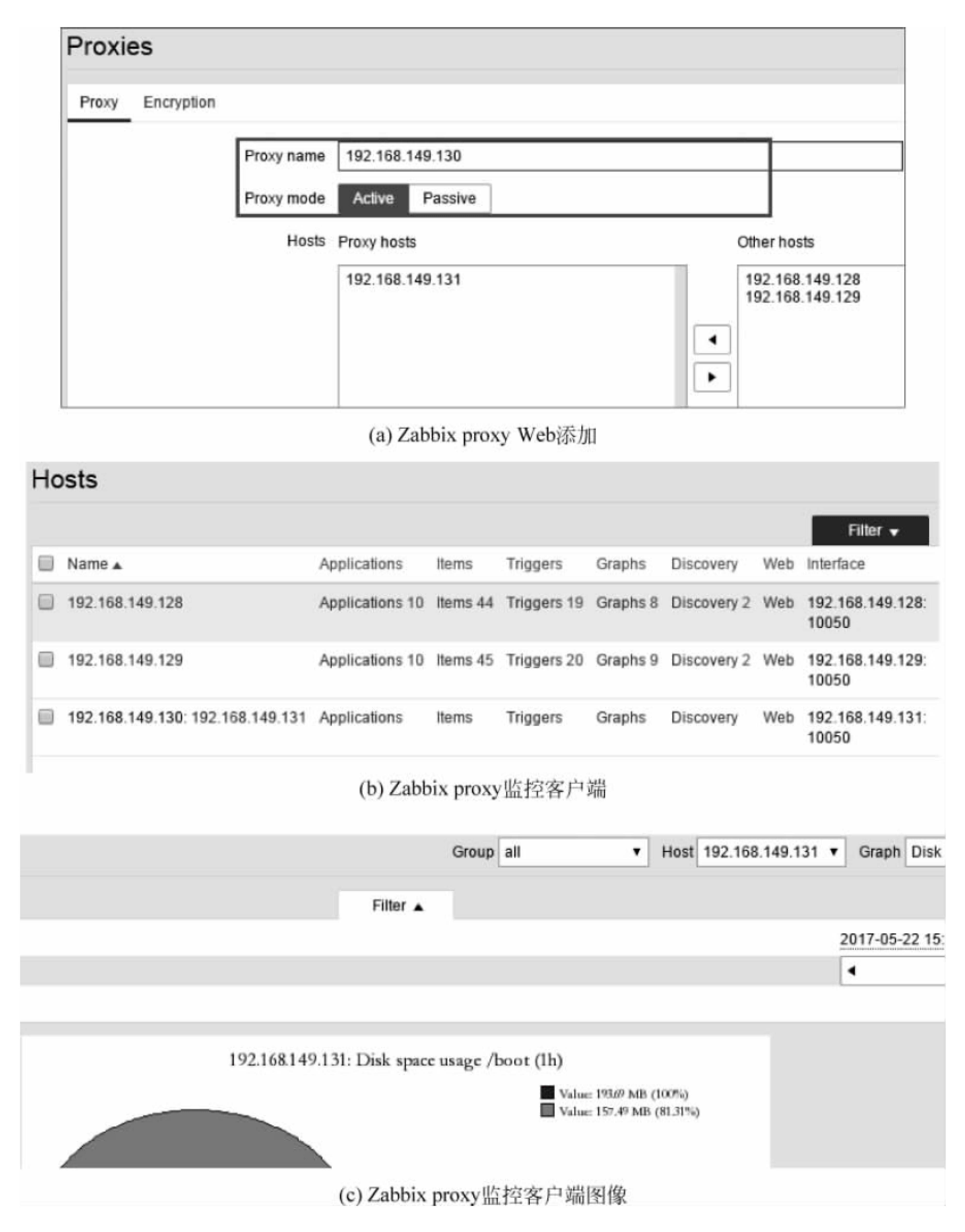
（5）将 zabbix_proxy.conf 配置文件中代码设置为如下代码：

```
Server = 192.168.149.128
Hostname = 192.168.149.130
LogFile = /tmp/zabbix_proxy.log
DBName = zabbix_proxy
DBUser = zabbix
DBPassword = 123456
Timeout = 4
LogSlowQueries = 3000
DataSenderFrequency = 30
HistoryCacheSize = 128MB
CacheSize = 128MB
```

（6）Zabbix 客户端安装 agent，同时配置 agent 端 server 设置为 proxy 服务器的 IP 地址或者主机名，zabbix_agentd.conf 配置文件代码如下：

```
LogFile = /tmp/zabbix_agentd.log
Server = 192.168.149.130
ServerActive = 192.168.149.130
Hostname  =  192.168.149.131
```

（7）在 Zabbix server Web 端添加 proxy，实现集中管理和分布式添加监控，如图 13-38 所示。



(a) Zabbix proxy Web添加

(b) Zabbix proxy监控客户端

(c) Zabbix proxy监控客户端图像

图 13-38　Zabbix server Web 端添加 proxy

13.13　Zabbix 微信报警

Zabbix 除了可以使用邮件报警之外，还可以通过多种方式把告警信息发送到指定人，例如短信、微信报警方式，越来越多的企业开始使用 Zabbix 结合微信作为主要的告警方式，

因为每个人每天都在使用微信，这样可以及时有效地把告警信息推送到接收人，方便对告警信息及时处理。Zabbix 微信报警设置方法及步骤如下：

（1）微信企业号注册。

企业号注册地址为 https://qy.weixin.qq.com/，填写企业注册信息，等待审核完，并且微信扫描登录企业公众号，如图 13-39 所示。



图 13-39 注册并登录微信企业公众号

（2）通讯录添加运维部门及人员。

登录新建的企业号，提前把企业成员信息添加到组织或者部门，需要填写手机号、微信号、邮箱，通过这样的方式让别人扫码关注企业公众号，以便于后面企业号推送消息给企业成员，如图 13-40 所示。

（3）在企业公众号中创建应用。

除了对个人添加微信报警之外，还可以添加不同管理组，接受同一个应用推送的消息，

成员账号，组织部门 ID，应用 agent ID，corp ID 和 secret，调用 API 接口需要用到这些信息，如图 13-41 所示。



(a) 登录企业公众号通讯录

(b) 添加企业成员信息

图 13-40　微信企业公众号通讯录



(a) 微信企业公众号创建应用(1)

图 13-41　微信企业公众号创建应用

应用logo

建议使用750*750，1M以内的jpg、png图片

应用名称

京峰运维监控报警

应用介绍（选填）

京峰运维监控报警，实现微信报警信息展示！
www.jfedu.net

可见范围

吴光科　黄小凯　添加

创建应用

(b) 微信企业公众号创建应用(2)

京峰运维监控报警

京峰运维监控报警

京峰运维监控报警，实现微信报警信息展示！www.jfedu.net

已启用

AgentId　1000004　编辑

Secret　PGEVed0I-myyCSuQV7wuB7kAEq63vKADW40XUjPMP4

可见范围　吴光科　黄小凯

(c) 微信企业公众号创建应用(3)

图 13-41 （续）

（4）获取企业 corp ID，单击企业公众号首页中“我的企业”即可看到，如图 13-42 所示。



图 13-42 微信企业公众号 corp ID

（5）微信接口调试，调用微信接口需要一个调用接口的凭证，Access Token 通过 corp ID 和 secret 可以获得 Access Token，微信企业号接口调试地址为 http://qydev.weixin.qq.com/debug，如图 13-43 所示。



图 13-43　微信企业公众号调试

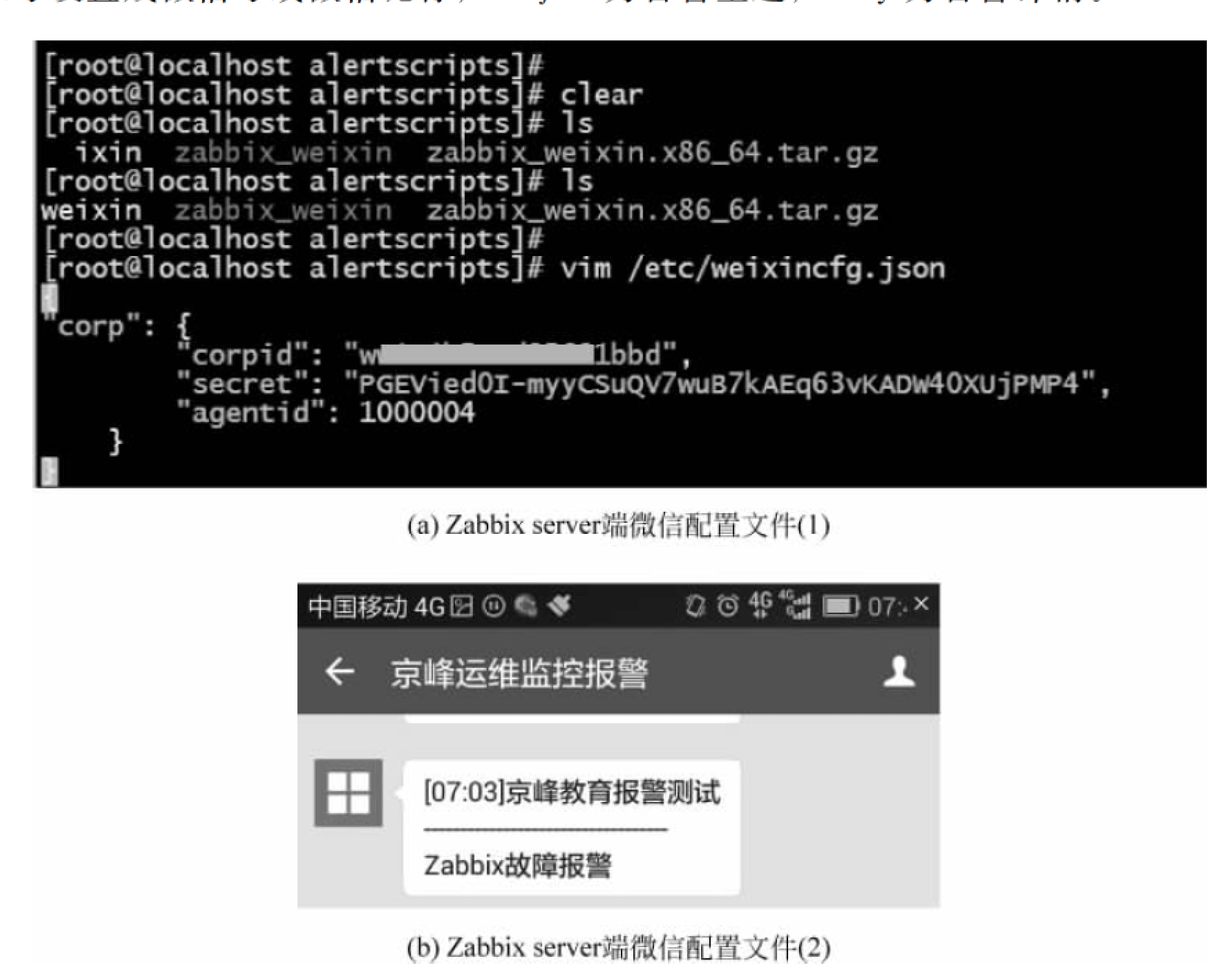
（6）获取微信报警工具，代码如下：

```
mkdir -p /usr/local/zabbix/alertscripts
cd /usr/local/zabbix/alertscripts
wget http://dl.cactifans.org/tools/zabbix_weixin.x86_64.tar.gz
tar zxvf zabbix_weixin.x86_64.tar.gz
mv zabbix_weixin/weixin
chmod o+x weixin
mv zabbix_weixin/weixincfg.json /etc/
rm -rf zxvf zabbix_weixin.x86_64.tar.gz
rm -rf zabbix_weixin/
```

修改/etc/ weixincfg. json 配置文件中 corp ID、secret、agent ID，并测试脚本发送信息，代码如下，如图 13-44 所示。

```
cd /usr/local/zabbix/alertscripts
./weixin wuguangke 京峰教育报警测试 Zabbix 故障报警
./weixin contact subject body
```

信息格式参数说明：contact 为你的微信账号，注意不是微信号，不是微信昵称，可以把用户账号设置成微信号或微信昵称；subject 为告警主题；body 为告警详情。



(a) Zabbix server端微信配置文件(1)

(b) Zabbix server端微信配置文件(2)

图 13-44　Zabbix server 端微信配置文件

（7）脚本调用设置。

Zabbix server 端设置脚本执行路径，编辑 zabbix_server. conf 文件，添加如下代码：

```
AlertScriptsPath = /usr/local/zabbix/alertscripts
```

（8）Zabbix Web 端配置，设置 Action 动作，并设置触发微信报警，如图 13-45 所示。

（9）配置 Media Types 微信脚本，依次选择 Administration→Media Types→Create Media Type，如图 13-46 所示，脚本加入 3 个参数：{ALERT. SENDTO}、{ALERT. SUBJECT}、{ALERT. MESSAGE}。

（10）配置接收微信信息的用户，依次选择 Administration→Users→Admin→Media，如图 13-47 所示。



(a) Zabbix server Action动作配置(1)

(b) Zabbix server Action动作配置(2)

(c) Zabbix server Action动作配置(3)

图 13-45 Zabbix server Action 动作配置

Name weixin_config
Type Script
Script name weixin
Script parameters
Parameter | Action
{ALERT.SENDTO} | Remove
{ALERT.SUBJECT} | Remove
{ALERT.MESSAGE} | Remove
Add
Enabled
Update Clone Delete Cancel

图 13-46　Zabbix server Media Types 配置

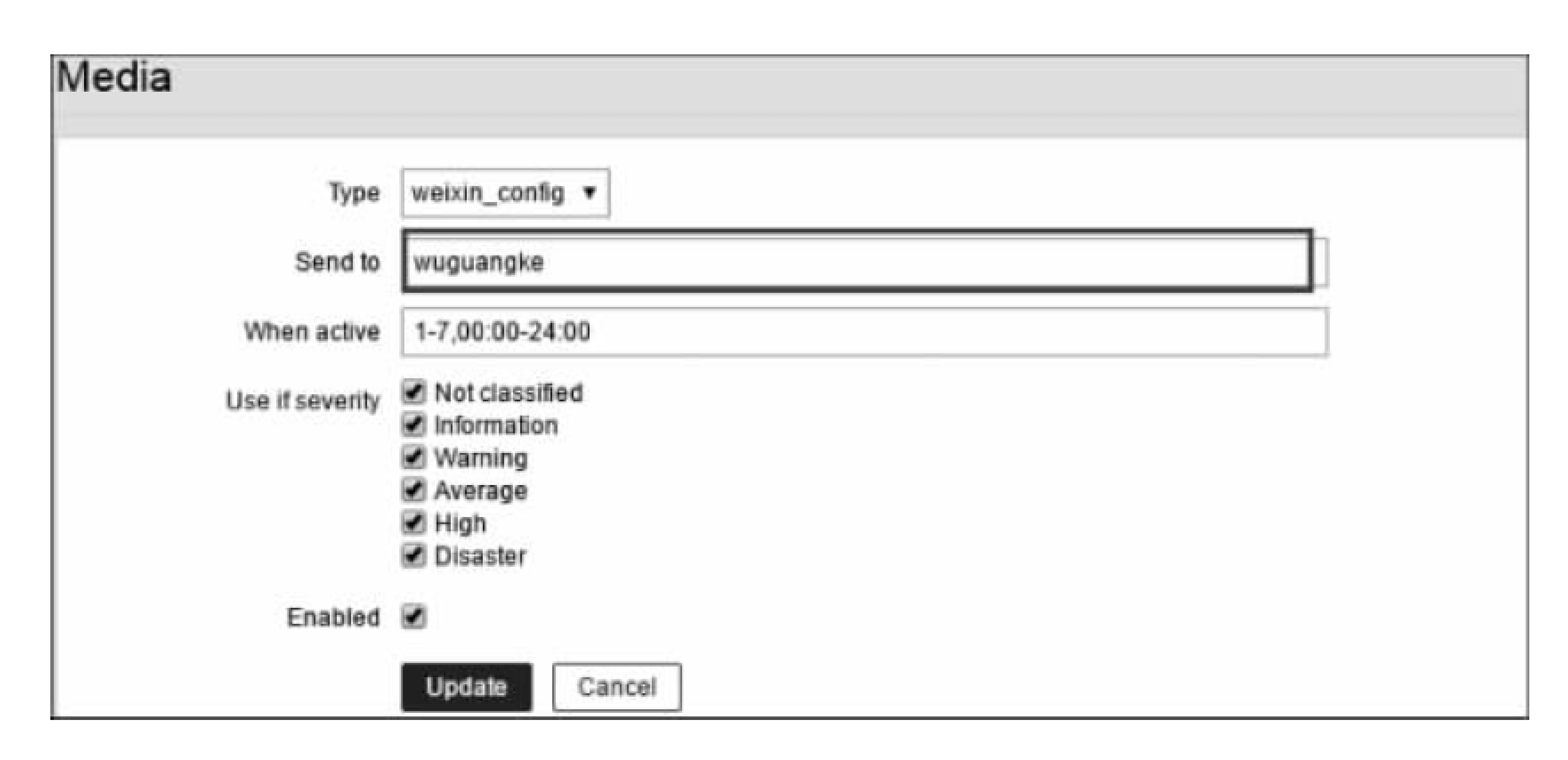


图 13-47　Zabbix server Users Media 配置

（11）微信报警信息测试，当磁盘容量剩余不足 20%时，会触发微信报警，如图 13-48 所示。



(a) Zabbix微信报警信息

图 13-48　Zabbix 微信报警信息测试



(b) Zabbix微信报警故障信息　　(c) Zabbix微信报警恢复信息

图 13-48　（续）

13.14　Zabbix 监控网站关键词

随着公司网站系统越来越多，不能通过人工每天手动去刷新网站来检查网站代码及页面是否被窜改，通过 Zabbix 监控可以实现自动去检查 Web 网站是否被窜改，例如监控某个网站页面中关键词 ATM 是否被修改，通过脚本监控的方法如下：

（1）agent 端编写 shell 脚本监控网站关键词，/data/sh/目录 shell 脚本内容如下，详情如图 13-49 所示。

```
#!/bin/bash
#2017 年 5 月 24 日 09:49:48
#by author jfedu.net
###################
WEBSITE = "http://192.168.149.131/"
NUM = 'curl - s $WEBSITE|grep - c "ATM"'
echo $NUM
```

```
[root@localhost sh]# cat check_http_word.sh
#!/bin/bash
#2017年5月24日09:49:48
#by author jfedu.net
###################
WEBSITE="http://192.168.149.131/"
NUM=`curl -s $WEBSITE|grep -c "ATM"`
echo $NUM
[root@localhost sh]#
[root@localhost sh]#
[root@localhost sh]# sh check_http_word.sh
1
[root@localhost sh]#
[root@localhost sh]#
```

图 13-49　Zabbix 客户端脚本内容

（2）在客户端 zabbix_agentd.conf 内容中加入如下代码，并重启 agentd 服务即可，如图 13-50 所示。

```
UserParameter = check_http_word, sh /data/sh/check_http_word.sh
```

```
[root@localhost sh]# cat /usr/local/zabbix/etc/zabbix_agentd.conf
LogFile=/tmp/zabbix_agentd.log
Server=192.168.149.128
ServerActive=192.168.149.128
Hostname = 192.168.149.131
UserParameter=check_http_word,sh /data/sh/check_http_word.sh
[root@localhost sh]#
[root@localhost sh]# pwd
/data/sh
[root@localhost sh]#
```

图 13-50　Zabbix 客户端脚本执行结果

（3）服务器端获取客户端的关键词 Key，输入 1，则表示 ATM 关键词存在，如果不为 1 则表示 ATM 关键词被窜改，代码如下：

```
/usr/local/zabbix/bin/zabbix_get -s 192.168.149.131 -k check_http_word
```

（4）Zabbix Web 端添加客户端的 Items 监控项，如图 13-51 所示。

Items

All hosts / 192.168.149.131　Enabled　ZBX SNMP JMX IPMI　Applications 10　Items 45　Triggers 19　Graphs 8　Dis

Name　HTTP Word monitor

Type　Zabbix agent

Key　check_http_word　Select

Host interface　192.168.149.131 : 10050

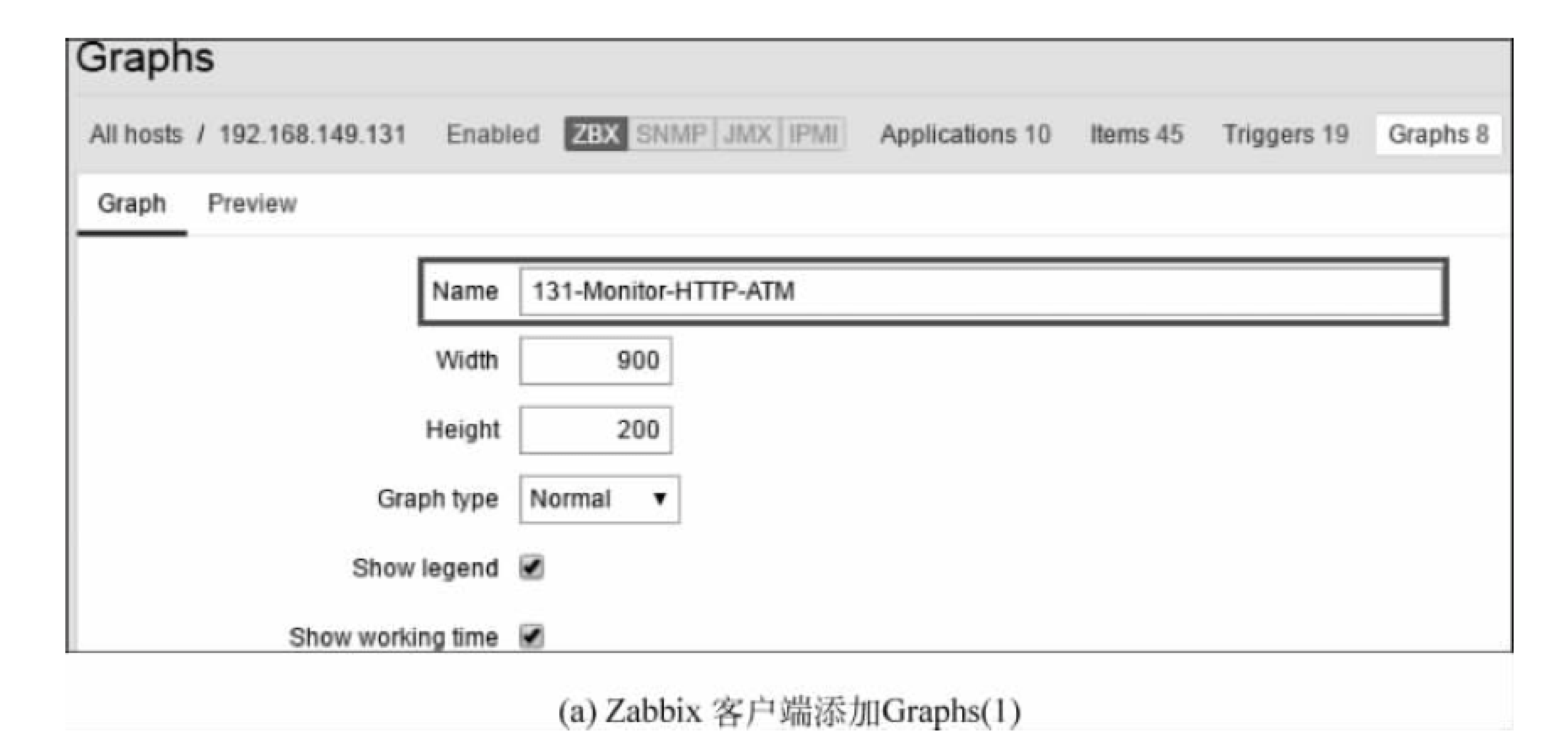
Type of information　Numeric (unsigned)

Data type　Decimal

Units

图 13-51　Zabbix 客户端 Key 添加

（5）创建 check_http_word 监控 Graphs 图像，如图 13-52 所示。



(a) Zabbix 客户端添加Graphs(1)

图 13-52　Zabbix 客户端添加 Graphs



(b) Zabbix 客户端添加Graphs(2)

图 13-52　（续）

（6）创建 check_http_word 触发器，如图 13-53 所示。

192.168.149.128/popup_trexpr.php
Item　192.168.149.131: HTTP Word monitor　Select
Function　Last (most recent) T value is NOT N
Last of (T)　Time
Time shift　Time
N　1
Insert　Cancel

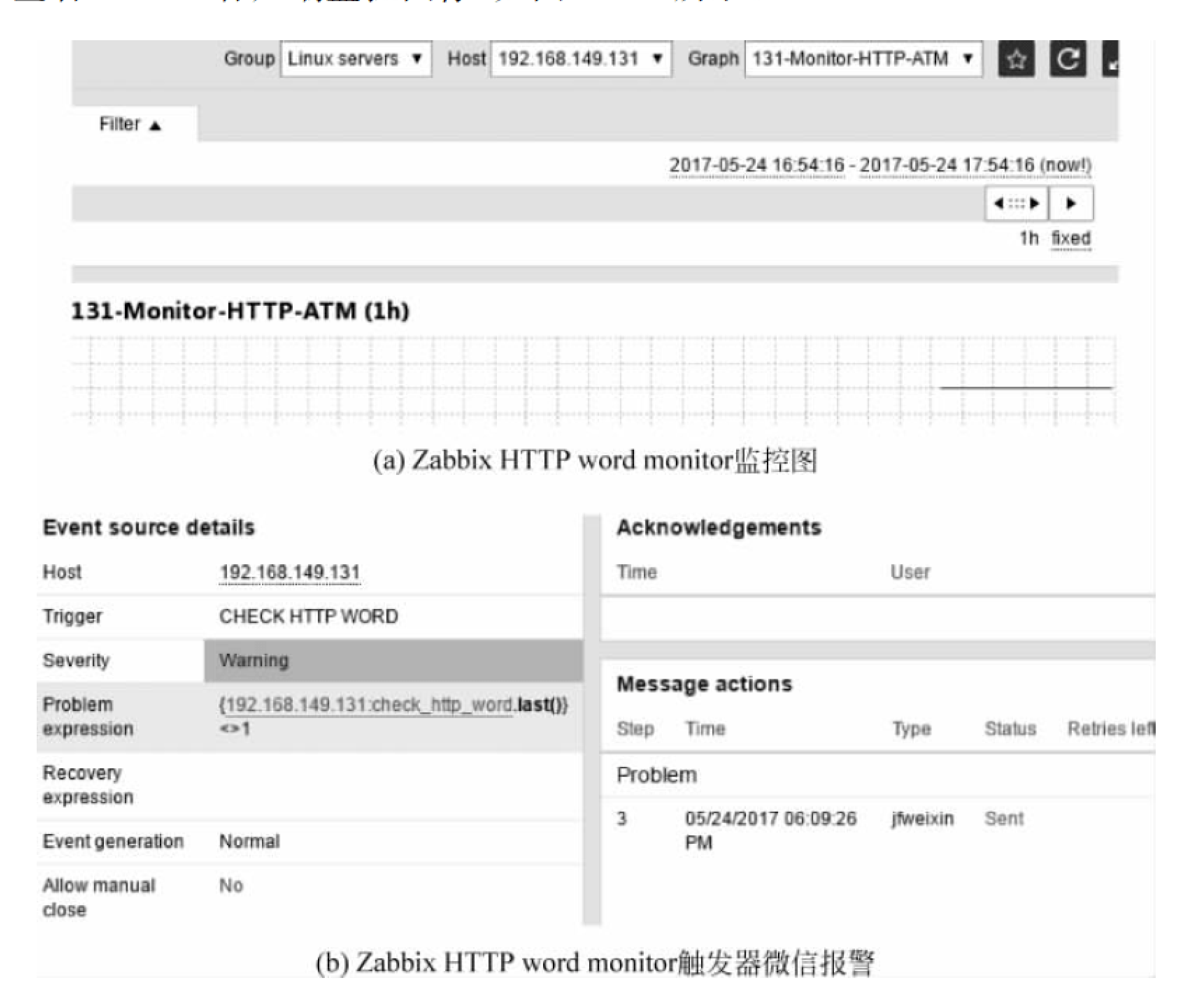
(a) Zabbix客户端创建触发器(1)

Trigger　Dependencies
Name　CHECK HTTP WORD
Severity　Not classified　Information　Warning　Average　High　Disaster
Expression　{192.168.149.131:check_http_word.last()}<>1　Add
Expression constructor
OK event generation　Expression　Recovery expression　None
PROBLEM event generation mode　Single　Multiple
OK event closes　All problems　All problems if tag values match

(b) Zabbix客户端创建触发器(2)

图 13-53　Zabbix 客户端创建触发器

（7）查看 Zabbix 客户端监控图像，如图 13-54 所示。



(a) Zabbix HTTP word monitor监控图

(b) Zabbix HTTP word monitor触发器微信报警

图 13-54　查看 Zabbix 客户端监控图像

除了使用如上 shell 脚本方式监控，还可以通过 Zabbix Web 界面配置 HTTP URL 监控，方法如下：依次选择 Configuration→Hosts→Web，创建 Web 监控场景，基于 Chrome 38.0 访问 HTTP Web 页面，如图 13-55 所示。



(a) Zabbix Web场景配置(1)

图 13-55　通过 Zabbix Web 界面配置 HTTP URL 监控

(b) Zabbix Web场景配置(2)

(c) Zabbix Web场景配置(3)

(d) Zabbix Web监控图(1)

(e) Zabbix Web监控图(2)

图 13-55　（续）

第 14 章 Nginx Web 服务器企业实战



万维网(world wide web,WWW)服务器,也称之为 Web 服务器,主要功能是提供网上信息浏览服务。目前主流的 Web 服务器软件包括 Apache、Nginx、Lighttpd、IIS、Resin、Tomcat、WebLogic、Jetty。

本章向读者介绍 Nginx 高性能 Web 服务器、Nginx 工作原理、安装配置及升级、Nginx 配置文件深入剖析、Nginx 虚拟主机、location 案例演示、Nginx rewirte 企业案例实战、HTTPS 安全 Web 服务器及 Nginx 高性能集群实战等内容。

14.1 Nginx Web 入门简介

Nginx(engine x)是一个高性能 HTTP、反向代理、IMAP、POP3、SMTP 服务器。Nginx 是由 Igor Sysoev 为俄罗斯访问量第二的 Rambler.ru 站点开发的,第一个公开版本发布于 2004 年 10 月 4 日。其源代码以类 BSD 许可证的形式发布,因它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。

由于 Nginx 的高性能、轻量级,目前越来越多的互联网企业开始使用 Nginx 做为 Web 服务器。据 Netcraft 统计,在 2017 年 4 月份,世界上最繁忙的网站中有 28.72%使用 Nginx 作为其服务器或者代理服务器。

Nginx 已经在众多流量很大的俄罗斯网站上使用了很长时间,这些网站包括 Yandex、Mail.Ru、VKontakte 以及 Rambler。目前互联网主流公司京东、360、百度、新浪、腾讯、阿里都在使用 Nginx 作为自己的 Web 服务器。

Nginx 特点是占有内存少,并发能力强,事实上 Nginx 的并发能力确实在同类型的网页服务器中表现较好。

Nginx 相对于 Apache 优点如下:

- 高并发响应性能非常好,官方 Nginx 处理静态文件并发 5w/s;
- 负载均衡及反向代理性能非常强;
- 系统内存和 CPU 占用率低;
- 可对后端服务进行健康检查;

- 支持 PHP CGI 方式和 FastCGI 方式；
- 可以作为缓存服务器、邮件代理服务器；
- 配置代码简洁且容易上手。

14.2　Nginx 工作原理

Nginx Web 服务器主要是由各种模块协同工作，模块从结构上分为核心模块、基础模块和第三方模块，其中三类模块分别如下：

- 核心模块：HTTP 模块、event 模块和 mail 模块等。
- 基础模块：HTTP access 模块、HTTP FastCGI 模块、HTTP proxy 模块和 HTTP rewrite 模块。
- 第三方模块：HTTP upstream request hash 模块、notice 模块和 HTTP access key 模块、limit_req 模块等。

Nginx 的模块从功能上分为如下三类。

- handlers(处理器模块)：此类模块直接处理请求，并进行输出内容和修改 headers 信息等操作，handlers 处理器模块一般只能有一个。
- filters(过滤器模块)：此类模块主要对其他处理器模块输出的内容进行修改操作，最后由 Nginx 输出。
- proxies(代理类模块)：此类模块是 Nginx 的 HTTP upstream 之类的模块，这些模块主要与后端一些服务比如 FastCGI 等进行交互，实现服务代理和负载均衡等功能。

Nginx 由内核和模块组成，其中内核的设计非常微小和简洁，完成的工作也非常简单，仅是通过查找配置文件将客户端的请求映射到一个 location block，而 location 是 Nginx 配置中的一个指令，用于访问的 URL 匹配，而 location 中所配置的每个指令将会启动不同的模块去完成相应的工作，如图 14-1 所示。

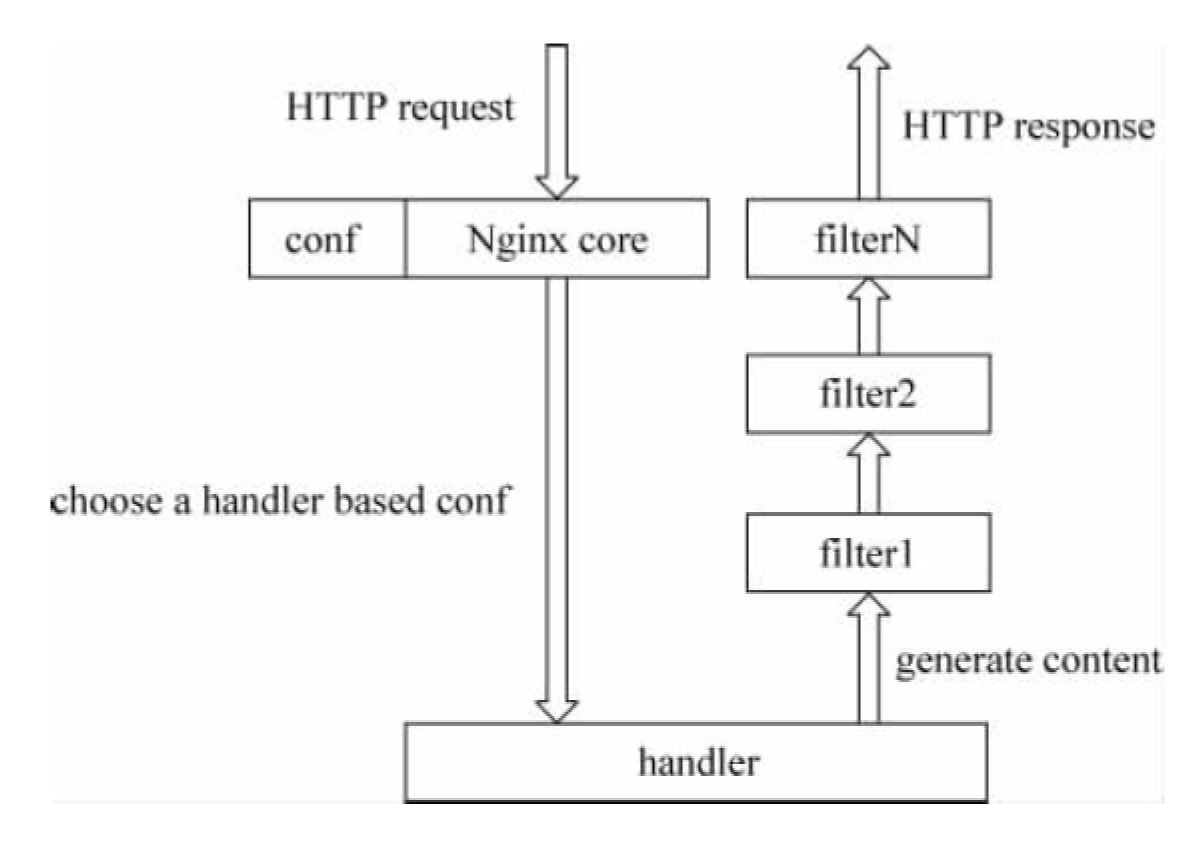


图 14-1　Nginx Web 工作流程图

Nginx 的高并发得益于其采用了 epoll 模型，与传统的服务器程序架构不同，epoll 是 Linux 内核 2.6 以后才出现的，Nginx 采用 epoll 模型，异步非阻塞，而 Apache 采用的是 select 模型。

select 模型的特点为 select 选择句柄的时候，是遍历所有句柄，也就是说句柄有事件响应时，select 需要遍历所有句柄才能获取到哪些句柄有事件通知，因此效率是非常低。

epoll 模型的特点为 epoll 对于句柄事件的选择不是遍历的，是事件响应的，就是句柄上事件来就马上选择出来，不需要遍历整个句柄链表，因此效率非常高。

Nginx 默认以 80 端口监听在服务器上，并且启动一个 master 进程，同时由 master 进程生成多个工作进程，当浏览器发起一个 HTTP 连接请求，每个进程都有可能处理这个连接。怎样保证同一时刻一个 HTTP 请求被一个工作进程处理呢？

首先每个 worker 进程都是从 master 进程 fork 出来，在 master 进程里面，建立好需要 listen 的 socket(listenfd)之后，会 fork 出多个 worker 进程。所有 worker 进程的 listenfd 会在新连接到来时变得可读，为保证只有一个进程处理该连接，所有 worker 进程在注册 listenfd 读事件前抢 accept_mutex，抢到互斥锁的那个进程注册 listenfd 读事件，在读事件里调用 accept 接受该连接。当一个 worker 进程在 accept 这个连接之后，就开始读取请求、解析请求、处理请求，产生数据后，再返回给客户端，最后才断开连接，这样形成一个完整的请求流程，如图 14-2 所示。

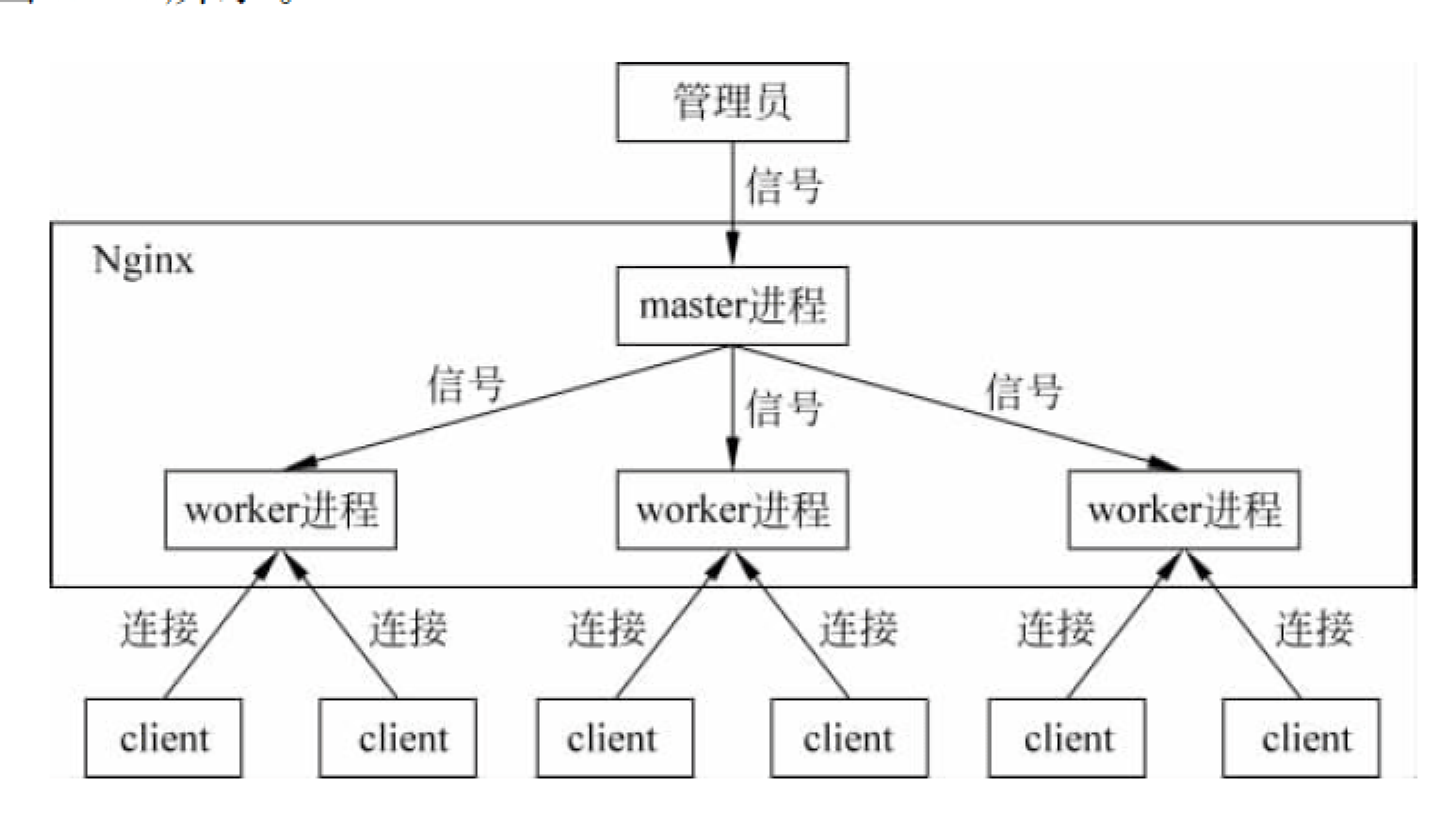


图 14-2　Nginx worker 进程工作原理

14.3　Nginx 安装配置

Nginx Web 安装时可以指定很多的模块，默认需要安装 rewrite 模块，需要系统有 PCRE 库，安装 PCRE 支持 rewrite 功能。以下为安装 Nginx Web 服务器的方法，注意 Nginx 整合 PCRE 库，需要指定 PCRE 源码目录，而不是 PCRE 编译完成之后的路径，否则会报错。代码如下：

```
#安装 PCRE 库支持
yum install pcre-devel pcre -y
```

```
#下载 Nginx 源码包
cd /usr/src
wget -c http://nginx.org/download/nginx-1.12.0.tar.gz
#解压 Nginx 源码包
tar -xzf nginx-1.12.0.tar.gz
#进入解压目录,然后 sed 修改 Nginx 版本信息为 JWS
cd nginx-1.12.0 ; sed -i -e 's/1.12.0//g' -e 's/nginx\//JWS/g' -e
's/"NGINX"/"JWS"/g' src/core/nginx.h
#预编译 Nginx
useradd www ; ./configure --user=www --group=www --prefix=/usr/local/nginx --with-
http_stub_status_module --with-http_ssl_module
#.configure 预编译成功后,执行 make 命令进行编译
make
#make 执行成功后,执行 make install 正式安装
make install
#至此 Nginx Web 服务器安装完毕
```

测试 Nginx 服务安装是否正确,同时启动 Nginx Web 服务,具体步骤如下:

- 检查 Nginx 配置文件是否正确,返回 OK 即正确。代码如下:

```
/usr/local/nginx/sbin/nginx -t
[root@localhost ~]# /usr/local/nginx/sbin/nginx -t
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
[root@localhost ~]#
```

- 然后启动 Nginx,执行命令/usr/local/nginx/sbin/nginx 按 Enter 键即可。查看进程是否已启动,代码如下:

```
[root@localhost ~]# ps -ef |grep nginx
nobody   5381 30285 0 May16 ?      09:04:31 nginx: worker process
root     30285     1 0 2017 ?    00:00:00 nginx: master process /usr/local/nginx/sbin/nginx
root     32260 32220 0 12:34 pts/0   00:00:00 grep nginx
[root@localhost ~]#
```

通过浏览器访问 Nginx 默认测试页面,如图 14-3 所示。

图 14-3　Nginx Web 浏览器访问

14.4　Nginx 管理及升级

Nginx Web 服务器安装完毕后,可以执行以下命令对其进行管理和维护,命令如下:

```
#查看 Nginx 进程
```

```
ps -ef|grep nginx
#平滑启动 Nginx
kill -HUP 'cat /var/run/nginx.pid'
#或者
nginx -s reload
#其中进程文件路径在配置文件 nginx.conf 中可以找到
#平滑启动的意思是在不停止 Nginx 的情况下,重启 Nginx,重新加载配置文件,启动新的工作线程,
#完美停止旧的工作线程
#完美停止 Nginx
kill -QUIT 'cat /var/run/nginx.pid'
#快速停止 Nginx
kill -TERM 'cat /var/run/nginx.pid'
#或者
kill -INT 'cat /var/run/nginx.pid'
#完美停止工作进程(主要用于平滑升级)
kill -WINCH 'cat /var/run/nginx.pid'
#强制停止 Nginx
pkill -9 nginx
#检查对 nginx.conf 文件的修改是否正确
nginx -t -c /etc/nginx/nginx.conf
#或者
nginx -t
#停止 Nginx 的命令
nginx -s stop
#或者
pkill nginx
#查看 Nginx 的版本信息
nginx -v
#查看完整的 Nginx 的配置信息
nginx -V
```

Nginx Web 软件定期更新，以下为将低版本升级或者将高版本降级的方法，一般分为四个步骤：软件下载、预编译、编译、配置，具体方法及代码如下：

```
wget http://www.nginx.org/download/nginx-1.4.2.tar.gz
#获取旧版本 Nginx 的 configure 选项
/usr/local/nginx/sbin/nginx -V
#编译新版本的 Nginx
tar -xvf nginx-1.4.2.tar.gz
cd nginx-1.4.2
./configure --prefix=/usr/local/nginx --user=www --group=www --with-http_stub_
status_module --with-http_ssl_module
make
#备份旧版本的 Nginx 可执行文件,复制新版本的 Nginx 可执行文件
mv /usr/local/nginx/sbin/nginx /usr/local/nginx/sbin/nginx.old
cp objs/nginx /usr/local/nginx/sbin/
#测试新版本 Nginx 是否正常
/usr/local/nginx/sbin/nginx -t
#平滑重启升级 Nginx
```

```
kill -QUIT 'cat /usr/local/nginx/log/nginx.oldbin'
#验证 Nginx 是否升级成功
/usr/local/nginx/sbin/nginx -V
#显示最新编译的版本信息即可
```

14.5 Nginx 配置文件优化一

学习 Nginx 服务的难点在于对配置文件的理解和优化，熟练掌握 Nginx 配置文件参数的含义可以更快地掌握 Nginx，以下为 nginx.conf 配置文件常用参数详解：

```
#定义 Nginx 运行的用户和用户组
user www www;
#启动进程，通常设置成和 CPU 的数量相等
worker_processes 8;
worker_cpu_affinity 00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000;
#为每个进程分配 CPU，上例中将 8 个进程分配到 8 个 CPU，当然可以写多个，或者将一个进程分配到
#多个 CPU
worker_rlimit_nofile 102400;
#该指令是当一个 Nginx 进程打开的最多文件描述符数目，理论值应该是最多打开文件数(ulimit -
#n)与 Nginx 进程数相除，但是 Nginx 分配请求并不是那么均匀，所以最好与 ulimit -n 的值保持
一致
#全局错误日志及 PID 文件
error_log /usr/local/nginx/logs/error.log;
#错误日志定义等级，[ debug | info | notice | warn | error | crit ]
pid        /usr/local/nginx/nginx.pid;
#工作模式及连接数上限
events {
use    epoll;
#epoll 是多路复用 I/O(I/O multiplexing)中的一种方式，但是仅用于 Linux 2.6 以上内核，可以大
#大提高 Nginx 的性能
worker_connections 102400;
#单个后台 worker process 进程的最大并发连接数(最大连接数 = 连接数 * 进程数)
multi_accept on;
#尽可能多地接受请求
}
#设定 HTTP 服务器，利用它的反向代理功能提供负载均衡支持
http {
#设定 MIME 类型，类型由 mime.type 文件定义
include        mime.types;
default_type   application/octet-stream;
#设定日志格式
access_log     /usr/local/nginx/log/nginx/access.log;
sendfile       on;
#sendfile 指令指定 Nginx 是否调用 sendfile 函数(zero copy 方式)来输出文件，对于普通应用必
#须设为 on
```

```
#如果用来进行下载等应用磁盘I/O重负载应用,可设置为 off,以平衡磁盘与网络 I/O 处理速度,降
#低系统的 uptime
#autoindex on;
#开启目录列表访问,合适下载服务器,默认关闭
tcp_nopush on;
#防止网络阻塞
keepalive_timeout 60;
#keepalive 超时时间,客户端到服务器端的连接持续有效时间,当出现对服务器的后继请求时,
#keepalive-timeout 功能可避免建立或重新建立连接
tcp_nodelay    on;
#提高数据的实时响应性
#开启 gzip 压缩
gzip on;
gzip_min_length 1KB;
gzip_buffers     4 16KB;
gzip_http_version 1.1;
gzip_comp_level 2;
#压缩级别大小,最大为 9,值越小,压缩后比例越小,CPU 处理更快,值越大,消耗 CPU 比较高
gzip_types        text/plain application/x-javascript text/css application/xml;
gzip_vary on;
client_max_body_size 10MB;
#允许客户端请求的最大单文件字节数
client_body_buffer_size 128KB;
#缓冲区代理缓冲用户端请求的最大字节数
proxy_connect_timeout 90;
#Nginx 跟后端服务器连接超时时间(代理连接超时)
proxy_send_timeout 90;
#后端服务器数据回传时间(代理发送超时)
proxy_read_timeout 90;
#连接成功后,后端服务器响应时间(代理接收超时)
proxy_buffer_size 4KB;
#设置代理服务器(Nginx)保存用户头信息的缓冲区大小
proxy_buffers 4 32KB;
#proxy_buffers 缓冲区,网页平均在 32KB 以下的话,这样设置
proxy_busy_buffers_size 64KB;
#高负荷下缓冲大小(proxy_buffers * 2)
#设定请求缓冲
large_client_header_buffers 4 4KB;
client_header_buffer_size 4KB;
#客户端请求头部的缓冲区大小,这个可以根据系统分页大小来设置,一般一个请求的头部大小不
#会超过 1KB
#不过由于一般系统分页都要大于 1KB,所以这里设置为分页大小。分页大小可以用命令 getconf
#PAGESIZE 取得
open_file_cache max=102400 inactive=20s;
#这个将为打开文件指定缓存,默认是没有启用的,max 指定缓存数量,建议和打开文件数一致,
#inactive 是指经过多长时间文件没被请求后删除缓存
open_file_cache_valid 30s;
```

```
#这个是指多长时间检查一次缓存的有效信息
open_file_cache_min_uses 1;
#open_file_cache 指令中的 inactive 参数时间内文件的最少使用次数，如果超过这个数字，文件描
#述符一直是在缓存中打开的，如上例，如果有一个文件在 inactive
#包含其他配置文件，如自定义的虚拟主机
include vhosts.conf;
```

14.6 Nginx 配置文件优化二

Nginx Web 默认发布静态页面，也可以均衡后端动态页面。用户发起 HTTP 请求，如果请求为静态页面，Nginx 直接处理并返回，如果请求的是动态页面，Nginx 收到请求之后会进行判断，转到后端服务器去处理。

Nginx 实现负载均衡需要基于 upstream 模块，同时需要设置 location proxy_pass 转发指令实现。

以下为 Ningx 应用负载均衡集群配置，根据后端实际情况修改即可，jfedu_www 为负载均衡模块的名称，可以任意指定，但必须跟 vhosts.conf、nginx.conf 虚拟主机的 proxy_pass 段保持一致，否则不能将请求转发至后端的服务器，weight 表示配置权重，在 fail_timeout 内检查 max_fails 次数，失败则剔除均衡。代码如下：

```
upstream jfedu_www {
        server 127.0.0.1:8080 weight = 1 max_fails = 2 fail_timeout = 30s;
        server 127.0.0.1:8081 weight = 1 max_fails = 2 fail_timeout = 30s;
    }
   #虚拟主机配置
    server {
        #侦听 80 端口
        listen       80;
        #定义使用 www.jfedu.net 访问
        server_name www.jfedu.net;
        #设定本虚拟主机的访问日志
        access_log logs/access.log main;
            root    /data/webapps/www;           #定义服务器的默认网站根目录位置
        index index.php index.html index.htm;    #定义首页索引文件的名称
        #默认请求
        location ~ /{
          root    /data/webapps/www;             #定义服务器的默认网站根目录位置
          index index.php index.html index.htm;  #定义首页索引文件的名称
          #以下是一些反向代理的配置
         proxy_next_upstream http_502 http_504 error timeout invalid_header;
         #如果后端的服务器返回 502、504、执行超时等错误，自动将请求转发到 upstream 负载均
         #衡池中的另一台服务器，实现故障转移
          proxy_redirect off;
          #后端的 Web 服务器可以通过 X - Forwarded - For 获取用户真实 IP
          proxy_set_header Host  $host;
          proxy_set_header X - Real - IP  $remote_addr;
```

```
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
           proxy_pass http://jfedu_www;          #请求转向后端定义的均衡模块
          }
          #定义错误提示页面
              error_page    500 502 503 504 /50x.html;
              location = /50x.html {
              root    html;
          }
          #配置 Nginx 动静分离,定义的静态页面直接从 Nginx 发布目录读取
          location ~ .*\.(html|htm|gif|jpg|jpeg|bmp|png|ico|txt|js|css)$
          {
              root /data/webapps/www;
              #expires 定义用户浏览器缓存的时间为 3 天,如果静态页面不常更新,可以设置更
              #长,这样可以节省带宽和缓解服务器的压力,在浏览器保存该类型文件的天数
              expires     3d;
          }
          #PHP 脚本请求全部转发到 FastCGI 处理.使用 FastCGI 默认配置
          location ~ \.php ${
              root /root;
              FastCGI_pass 127.0.0.1:9000;
              FastCGI_index index.php;
              FastCGI_param SCRIPT_FILENAME /data/webapps/www $FastCGI_script_name;
              include FastCGI_params;
          }
          #设定查看 Nginx 状态的地址
          location /NginxStatus {
              stub_status on;
          }
        }
    }
```

通过 expires 参数设置，可以在浏览器缓存静态文件，从而减少用户与服务器之间的请求和流量。具体 expires 定义是给一个资源设定一个过期时间，浏览器无须去服务端下载资源，直接通过浏览器自身确认是否过期即可，所以不会产生额外的流量。

如果静态文件不常更新，expires 可以设置为 30d，表示在这 30 天之内再次访问该静态文件，浏览器会发送一个 HTTP 请求，会比对服务器该文件最后更新时间是否有变化，如果没有变化，则不会从服务器抓取，返回 HTTP 状态码 304，如果有修改，则直接从服务器重新下载，返回 HTTP 状态码 200。

14.7 Nginx 虚拟主机实战

在真实的企业服务器环境中，为了充分利用服务器的资源，单台 Nginx Web 服务器同时会配置 N 个网站，也可称之为配置 N 个虚拟域名的主机，即多个域名对应同一个 80 端口。

在 Nginx.conf 中加入 server 代码，Nginx 虚拟主机配置代码如下：

```
worker_processes 1;
events {
    worker_connections 1024;
}
http {
    include         mime.types;
    default_type    application/octet-stream;
    sendfile        on;
    keepalive_timeout 65;
#virtual hosts config 2017/5/18
server {
        listen      80;
        server_name www.jf1.com;
        access_log logs/jf1.access.log;
        location / {
            root    html/jf1;
            index   index.html index.htm;
        }
}
server {
        listen      80;
        server_name www.jf2.com;
        access_log logs/jf2.access.log;
        location / {
            root    html/jf2;
            index   index.html index.htm;
        }

 }
}
```

创建两个不同的目录 mkdir -p /usr/local/nginx/html/{jf1,jf2}，然后分别在两个目录创建两个不同的 index.html 网站页面即可。通过 Windows 客户端配置 hosts 绑定 IP 与两个域名的对应关系，在 IE 浏览器访问测试效果，如图 14-4 所示。

www.jf1.com Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.

(a) Nginx虚拟主机www.jf1.com

图 14-4　IE 浏览器访问测试效果

www.jf2.com Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.

(b) Nginx虚拟主机www.jf2.com

图 14-4 （续）

14.8 Nginx location 深入剖析

Nginx 由内核和模块组成，其中内核的设计非常微小和简洁，完成的工作也非常简单，仅是通过查找配置文件将客户端的请求映射到一个 location block，而 location 是 Nginx 配置中的一个指令，用于访问的 URL 匹配，而在这个 location 中所配置的每个指令将会启动不同的模块去完成相应的工作。

默认 nginx.conf 配置文件中至少存在一个 location /，即表示客户端浏览器请求的 URL 为域名＋“/”，如果 location /newindex/，则表示客户端浏览器请求的 URL 为域名＋“/newindex/”。常见 location 匹配 URL 的方式如下：

- ＝：字面精确匹配。
- ^～：最大前缀匹配。
- /：不带任何前缀。
- ～：大小写相关的正则匹配。
- ～*：大小写无关的正则匹配。
- @：location 内部重定向的变量。

其中 location ＝、^～、/属于普通字符串匹配，location ～、～* 属于正则表达式匹配，location 优先级与其在 nginx.conf 配置文件中的先后顺序无关。

location ＝精确匹配会第一个被处理，如果发现精确匹配，Nginx 则停止搜索其他任何 location 的匹配。

普通字符匹配，正则表达式规则和完整 URL 规则将被优先查询匹配，^～为最大前缀匹配，如果匹配到该规则，Nginx 则停止搜索其他任何 location 的匹配，否则 Nginx 会继续处理其他 location 指令。

正则匹配“～”和“～*”，如果找到相应的匹配，则 Nginx 停止搜索其他任何 location 的匹配；当没有正则表达式或者没有正则表达式被匹配的情况下，那么匹配程度最高的逐字匹配指令会被使用。

location 规则匹配优先级总结如下：

（location ＝）＞（location 完整路径）＞（location ^～路径）＞（location ～或～* 正

则顺序）＞（location 部分起始路径）＞（locaton/）

以下为 Nginx location 规则案例演示：

```
location = / {
  [ configuration L1 ]
  #只会匹配/,优先级比 location /低
}
location = /index.html {
  [ configuration L2 ]
#只会匹配/index.html,优先级最高
}
location / {
  [ configuration L3 ]
  #匹配任何请求,因为所有请求都是以"/"开始
  #但是更长字符匹配或者正则表达式匹配会优先匹配,优先级最低
}
location = /images/ {
  [ configuration L4 ]
  #匹配任何以/images/开始的请求,并停止匹配其他 location
}
location ~* \.(html|txt|gif|jpg|jpeg)${
  [ configuration L5]
  # 匹配以 html、txt、gif、jpg、jpeg 结尾的 URL 文件请求
  # 但是所有/images/目录的请求将由 [configuration L4]处理
}
```

浏览器发起 HTTP request URI 案例与 location 规则案例匹配如下：

- / —>匹配 configuration L3；
- /index.html 匹配 configuration L2；
- /images/匹配 configuration L4；
- /images/logo.png 匹配 configuration L4；
- /img/test.jpg 匹配 configuration L5。

企业生产环境中无须在 nginx.conf 配置文件中同时添加 5 种规则匹配，以下为企业生产环境 Nginx location 部分配置代码：

```
location /
{
    root /var/www/html/;
    expires      60d;
}
location ~ .*\.(gif|jpg|jpeg|bmp|png|ico|txt|js|css)$
{
    root /var/www/html/;
    expires      60d;
}
```

```
location ~ .*\.(jsp|php|cgi|do)$
{
    root /var/www/html/;
    proxy_pass http://linux_web;
    proxy_http_version 1.1;
    proxy_set_header Connection "";
    proxy_set_header Host  $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
location =/newindex.html
{
    root /var/www/newwww/;
    expires      60d;
}
```

14.9 企业实战 Nginx 动静分离架构

Nginx 动静分离简单来说是把动态跟静态请求分开，不能理解成只是单纯地把动态页面和静态页面物理分离。严格意义上说应该是动态请求跟静态请求分开，可以理解成使用 Nginx 处理静态页面，Tomcat、Resin、PHP、ASP 处理动态页面。

动静分离从实现角度来讲大致分为两种：一种是纯粹地把静态文件独立成单独的域名，放在独立的服务器上，也是目前主流推崇的方案；另外一种方法就是动态跟静态文件混合在一起发布，通过 Nginx 来分开。

Nginx 线上 Web 服务器动静分离及 nginx.conf 完整配置文件代码如下：

```
user www www;
worker_processes 8;
worker_cpu_affinity 00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000;
pid /usr/local/nginx/nginx.pid;
worker_rlimit_nofile 102400;
events
{
use epoll;
worker_connections 102400;
}
http
{
  include       mime.types;
  default_type  application/octet-stream;
  FastCGI_intercept_errors on;
  charset utf-8;
  server_names_hash_bucket_size 128;
```

```
  client_header_buffer_size 4KB;
  large_client_header_buffers 4 32KB;
  client_max_body_size 300MB;
  sendfile on;
  tcp_nopush      on;
  keepalive_timeout 60;
  tcp_nodelay on;
  client_body_buffer_size     512KB;
  proxy_connect_timeout       5;
  proxy_read_timeout          60;
  proxy_send_timeout          5;
  proxy_buffer_size           16KB;
  proxy_buffers               4 64KB;
  proxy_busy_buffers_size     128KB;
  proxy_temp_file_write_size  128KB;
  gzip on;
  gzip_min_length             1KB;
  gzip_buffers                4 16KB;
  gzip_http_version           1.1;
  gzip_comp_level             2;
  gzip_types        text/plain application/x-javascript text/css application/xml;
  gzip_vary on;
log_format main '$remote_addr - $remote_user [ $time_local] " $request" '
              '$status $body_bytes_sent " $http_referer" '
              '" $http_user_agent" $request_time';
upstream jvm_web1 {
    server   192.168.149.130:8080 weight=1 max_fails=2 fail_timeout=30s;
    server   192.168.149.130:8081 weight=1 max_fails=2 fail_timeout=30s;
}
include vhosts.conf;
}
```

以下为 vhosts.conf 配置文件中的内容：

```
server
  {
    listen        80;
    server_name www.jf1.com;
    index index.jsp index.html index.htm;
    root /data/webapps/www1;
    location /
    {
         proxy_next_upstream http_502 http_504 error timeout invalid_header;
         proxy_set_header Host $host;
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
         proxy_pass http://jvm_web1;
```

```
    }
    location ~ . * \.(php|jsp|cgi|shtml)?$
    {
         proxy_set_header Host $host;
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
         proxy_pass http://jvm_web1;
    }
    location ~ . * \.(html|htm|gif|jpg|jpeg|bmp|png|ico|txt|js|css)$
    {
        root /data/webapps/www1;
        expires     30d;
    }
        access_log /data/logs/jvm_web1/access.log main;
        error_log  /data/logs/jvm_web1/error.log crit;
}
```

配置文件代码中 location ~ . * \.(php|jsp|cgi|shtml)表示匹配动态页面请求，然后将请求 proxy_pass 到后端服务器，而 location ~ . * \.(html|htm|gif|jpg|jpeg |ico|txt|js|css)表示匹配静态页面请求本地返回。

检查 Nginx 配置是否正确即可，然后测试动静分离是否成功，在 192.168.149.130 服务器启动 8080、8081 Tomcat 服务或者 LAMP 服务，删除后端 Tomcat 或者 LAMP 服务器上的某个静态文件，测试是否能访问该文件，如果可以访问，则说明静态资源通过 Nginx 直接返回了，如果不能访问，则证明动静分离不成功。

14.10 企业实战 LNMP 高性能服务器

公共网关接口(common gateway interface，CGI)，是 HTTP 服务器与本机或者其他机器上的程序进行通信的一种工具，其程序需运行在网络服务器上。

CGI 可以用任何一种语言编写，只要这种语言具有标准输入、输出和环境变量，如 PHP、Perl、TCL 等。

FastCGI 为 Web 服务器与处理程序之间通信的一种协议(App server 和 Web server 之间的通信协议)，是 CGI 的改进方案。CGI 程序反复加载是 CGI 性能低下的主要原因，如果 CGI 程序保持在内存中并接受 FastCGI 进程管理器调度，则可以提供良好的性能、伸缩性、Fail-Over 特性等。FastCGI 是常驻型的 CGI，它可以一直运行，在请求到达时，不会花费时间去 fork 一个进程来处理。

FastCGI 是语言无关的、可伸缩架构的 CGI 开放扩展，将 CGI 解释器进程保持在内存中，以此获得较高的性能。FastCGI 是一个协议，PHP-FPM 实现了这个协议，PHP-FPM 的 FastCGI 协议需要有进程池，PHP-FPM 实现的 FastCGI 进程叫 PHP-CGI，所以 PHP-FPM 其实是它自身的 FastCGI 或 PHP-CGI 进程管理器，如图 14-5 所示。

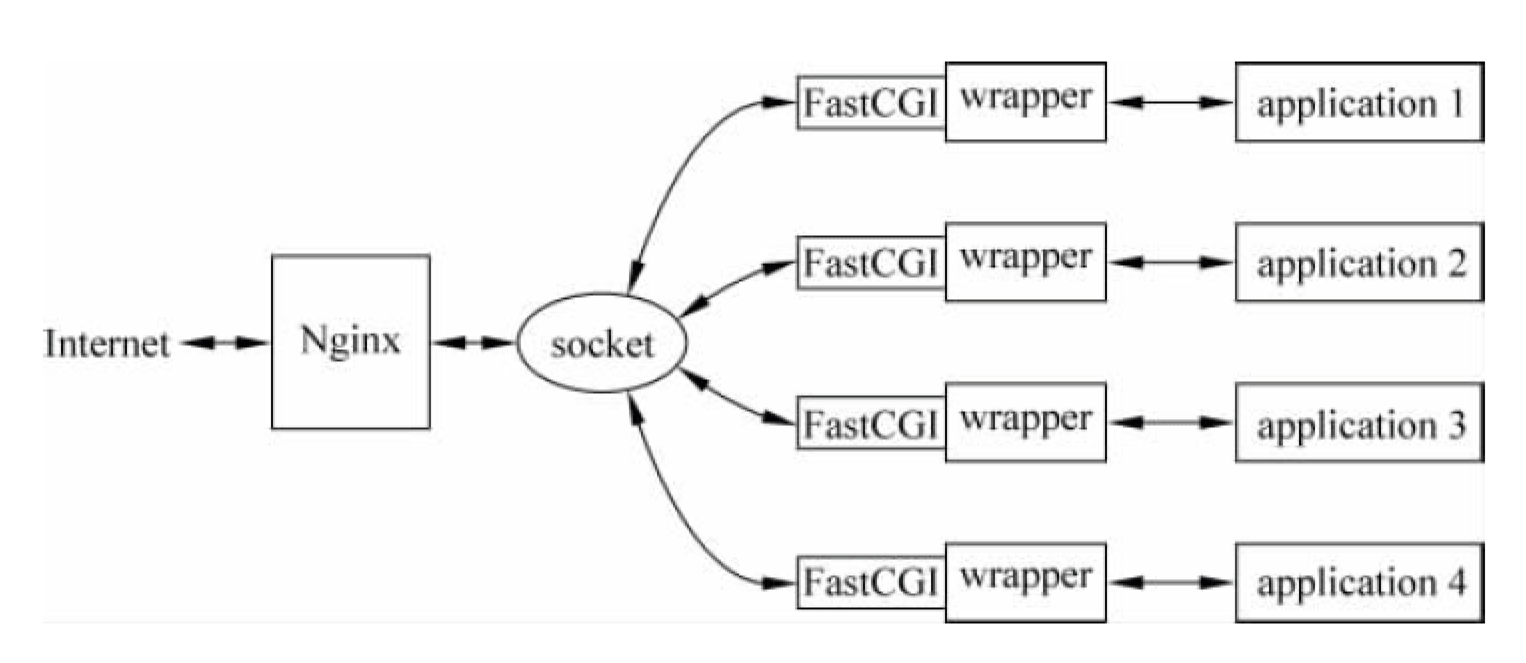


图 14-5　Nginx＋FastCGI 通信原理图

企业级 LNMP(Nginx＋PHP(FastCGI)＋MySQL)主流架构配置方法如下，分别安装 Nginx、MySQL、PHP 服务，具体步骤如下：

(1) Nginx 安装配置，代码如下：

```
wget -c http://nginx.org/download/nginx-1.12.0.tar.gz
tar -xzf nginx-1.12.0.tar.gz
cd nginx-1.12.0
useradd www ; ./configure --user=www --group=www --prefix=/usr/local/nginx --with-
http_stub_status_module --with-http_ssl_module
make
make install
```

(2) MySQL 安装配置，代码如下：

```
yum install cmake ncurses-devel ncurses -y
wget http://down1.chinaunix.net/distfiles/mysql-5.5.20.tar.gz
tar -xzf mysql-5.5.20.tar.gz
cd mysql-5-5.20
cmake . -DCMAKE_INSTALL_PREFIX=/usr/local/mysql55 \
-DMYSQL_UNIX_ADDR=/tmp/mysql.sock \
-DMYSQL_DATADIR=/data/mysql \
-DSYSCONFDIR=/etc \
-DMYSQL_USER=mysql \
-DMYSQL_TCP_PORT=3306 \
-DWITH_XTRADB_STORAGE_ENGINE=1 \
-DWITH_INNOBASE_STORAGE_ENGINE=1 \
-DWITH_PARTITION_STORAGE_ENGINE=1 \
-DWITH_BLACKHOLE_STORAGE_ENGINE=1 \
-DWITH_MYISAM_STORAGE_ENGINE=1 \
-DWITH_READLINE=1 \
-DENABLED_LOCAL_INFILE=1 \
-DWITH_EXTRA_CHARSETS=1 \
```

```
-DDEFAULT_CHARSET=utf8 \
-DDEFAULT_COLLATION=utf8_general_ci \
-DEXTRA_CHARSETS=all \
-DWITH_BIG_TABLES=1 \
-DWITH_DEBUG=0
make
make install
```

(3) PHP 安装配置，代码如下：

```
wget http://museum.php.net/php5/php-5.3.10.tar.gz
yum -y install gd curl curl-devel libjpeg libjpeg-devel libpng libpng-devel freetype
freetype-devel libxml2 libxml2-devel
cd php-5.3.10
./configure --prefix=/usr/local/php5 --enable-fpm --enable-debug --with-gd --with
-jpeg-dir --with-png-dir --with-freetype-dir --enable-mbstring --with-curl --
with-mysql=/usr/local/mysql5/ --with-mysqli=/usr/local/myql5/bin/mysql_config --
with-config-file-path=/usr/local/php5/etc
make
make install
cp php.ini-development   /usr/local/php5/etc/php.ini
cp /usr/local/php5/etc/php-fpm.conf.default /usr/local/php5/etc/php-fpm.conf
/usr/local/php5/sbin/php-fpm
cp sapi/fpm/init.d.php-fpm /etc/init.d/php-fpm
```

(4) Nginx 配置文件配置，代码如下：

```
server {
 include port.conf;
 server_name www.jfedu.net jfedu.net;
  location / {
 index index.html index.php;
 root /usr/local/nginx/html;
 }
 location ~ \.php${
           root           html;
           FastCGI_pass   127.0.0.1:9000;
           FastCGI_index  index.php;
           FastCGI_param  SCRIPT_FILENAME html $FastCGI_script_name;
           include        FastCGI_params;
        }
}
```

(5) 测试 LNMP 架构测试，创建 index.php 测试页面，如图 14-6 所示。

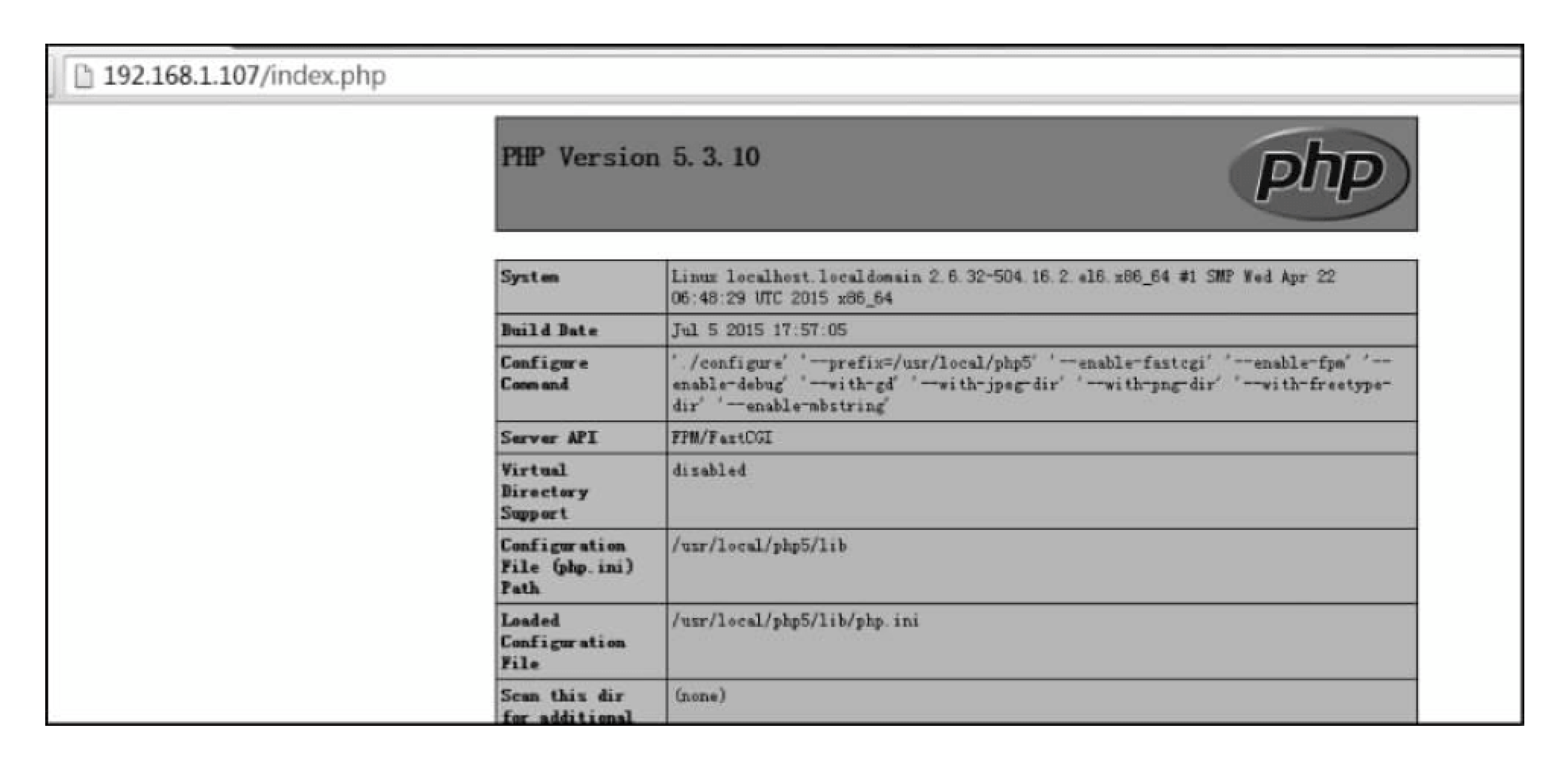


图 14-6 LNMP 企业实战测试页面

14.11 Nginx rewrite 规则详解

rewirte 规则也称为规则重写，主要功能是实现浏览器访问 HTTP URL 的跳转，其正则表达式是基于 Perl 语言。通常而言，几乎所有的 Web 服务器均可以支持 URL 重写。rewrite URL 规则重写的用途如下：

- 对搜索引擎优化(search engine optimization，SEO)友好，利于搜索引擎抓取网站页面；
- 隐藏网站 URL 真实地址，浏览器显示更加美观；
- 网站变更升级，可以基于 rewrite 临时重定向到其他页面。

Nginx rewrite 规则使用中有 3 个概念，分别是 rewrite 结尾标识符、rewrite 规则常用表达式、Nginx rewrite 变量，3 个概念的详解如下：

(1) Nginx rewrite 结尾标识符，用于 rewrite 规则末尾，表示规则的执行属性，详解如下：

- last：相当于 Apache 里的(L)标记，表示完成 rewrite 匹配。
- break：本条规则匹配完成后，终止匹配，不再匹配后面的规则。
- redirect：返回 302 临时重定向，浏览器地址会显示跳转后的 URL 地址。
- permanent：返回 301 永久重定向，浏览器地址栏会显示跳转后的 URL 地址。

其中 last 和 break 用来实现 URL 重写时，浏览器地址栏 URL 地址不变。

(2) Nginx rewrite 规则常用表达式，主要用于匹配参数、字符串及过滤设置，详解如下：

- .：匹配任何单字符。
- [word]：匹配字符串 word。

- [^word]：不匹配字符串 word。
- jfedu|jfteach：可选择的字符串 jfedu|jfteach。
- ?：匹配 0～1 个字符。
- *：匹配 0 到多个字符。
- +：匹配 1 到多个字符。
- ^：字符串开始标志。
- $：字符串结束标志。
- \n：转义符标志。

(3) Nginx rewrite 变量，常用于匹配 HTTP 请求头信息、浏览器主机名、URL 等，具体内容如下：

HTTP headers：HTTP_USER_AGENT，HTTP_REFERER，HTTP_COOKIE，HTTP_HOST，HTTP_ACCEPT。

connection & request：REMOTE_ADDR，QUERY_STRING。

server internals：DOCUMENT_ROOT，SERVER_PORT，SERVER_PROTOCOL。

system stuff：TIME_YEAR，TIME_MON，TIME_DAY。

详解如下：

- HTTP_USER_AGENT：用户使用的代理，例如浏览器。
- HTTP_REFERER：告知服务器，从哪个页面来访问的。
- HTTP_COOKIE：客户端缓存，主要用于存储用户名和密码等信息。
- HTTP_HOST：匹配服务器 servername 域名。
- HTTP_ACCEPT：客户端的浏览器支持的 MIME 类型。
- REMOTE_ADDR：客户端的 IP 地址。
- QUERY_STRING：URL 中访问的字符串。
- DOCUMENT_ROOT：服务器发布目录。
- SERVER_PORT：服务器端口。
- SERVER_PROTOCOL：服务器端协议。
- TIME_YEAR：年。
- TIME_MON：月。
- TIME_DAY：日。

(4) Nginx rewrite 以下配置均配置在 nginx.conf 或者 vhosts.conf 中，企业中常用的 Nginx rewrite 案例如下：

- 将 jfedu.net 跳转至 www.jfedu.net，代码如下：

```
if ( $host = 'jfedu.net' ) {
        rewrite ^/(.*)$http://www.jfedu.net/$1 permanent;
}
```

- 访问 www.jfedu.net 跳转 www.test.com/new.index.html，代码如下：

```
rewrite   ^/$http://www.test.com/index01.html permanent;
```

- 访问/jfedu/test01/跳转至/newindex.html，浏览器地址不变，代码如下：

```
rewrite   ^/jfedu/test01/$     /newindex.html     last;
```

- 多域名跳转到 www.jfedu.net，代码如下：

```
if ( $host != 'www.jfedu.net') {
rewrite ^/(.*)$http://www.jfedu.net/$1 permanent;
}
```

- 访问文件和目录不存在跳转至 index.php，代码如下：

```
if ( ! -e $request_filename )
{
rewrite ^/(.*)$/index.php last;
}
```

- 目录对换/xxxx/123456⇒/xxxx? id=123456，代码如下：

```
rewrite    ^/(.+)/(\d+)     /$1?id= $2       last;
```

- 判断浏览器 user agent 跳转，代码如下：

```
if( $http_user_agent ~ MSIE)
{
rewrite ^(.*)$/ie/$1 break;
}
```

- 禁止访问以.sh，.flv，.mp3 为文件后缀名的文件，代码如下：

```
location ~ .*\.(sh|flv|mp3)$
{
    return 403;
}
```

- 将移动用户访问跳转至移动端，代码如下：

```
if (  $http_user_agent ~* "(Android)|(iPhone)|(Mobile)|(WAP)|(UCWEB)" )
{
rewrite ^/$      http://m.jfedu.net/       permanent;
}
```

- 匹配 URL 访问字符串跳转，代码如下：

```
if ( $args ~* tid=13){
 return 404;
}
```

- 访问/10690/jfedu/123 跳转至/index. php? tid/10690/items＝123，[0-9]表示任意一个数字，＋表示多个，(. ＋)表示任何多个字符，代码如下：

```
rewrite   ^/([0-9]+)/jfedu/(.+)$     /index.php?tid/$1/items=$2     last;
```

14.12 Nginx Web 日志分析

在企业服务器运维中，当 Nginx 服务器正常运行后，SA 会经常密切关注 Nginx 的访问日志，发现有异常的日志信息需要进行及时处理。

Nginx 默认日志路径/usr/local/nginx/logs/，其中包含访问日志 access. log 和错误记录日志 error. log，查看 Nginx 访问日志 cat /usr/local/nginx/logs/access. log | more，如图 14-7 所示。

```
114.252.151.43 - - [26/May/2017:00:12:49 +0800] "POST /forum.php?mod=
id=69&extra=&topicsubmit=yes HTTP/1.1" 302 0 "http://bbs.jfedu.net/fo
=newthread&fid=69&referer=http%3A//bbs.jfedu.net/forum.php%3Fmod%3Dfo
"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/5.
50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .
016
82.114.92.33 - - [26/May/2017:00:12:50 +0800] "GET /home.php?mod=spac
&from=space HTTP/1.0" 200 16219 "http://bbs.jingfengjiaoyu.com/" "Moz
1; WOW64; rv:35.0) Gecko/20100101 Firefox/35.0"  0.027
68.180.231.61 - - [26/May/2017:00:12:51 +0800] "GET /web/user/202 HTT
zilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ys
82.114.92.33 - - [26/May/2017:00:12:52 +0800] "GET /member.php?mod=re
00 "http://bbs.jingfengjiaoyu.com/" "Mozilla/5.0 (Windows NT 6.1; WOW
0101 Firefox/35.0"  0.025
111.201.200.184 - - [26/May/2017:00:12:52 +0800] "GET /member.php?mod
 5693 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64;
```

图 14-7　Nginx 访问日志信息

Nginx 访问日志打印的格式可以自定义，例如 Nginx 日志打印格式配置，log_format 用来设置日志格式，name 为模块名，type 为日志类型，可以配置多个日志模块，分别供不同的虚拟主机日志记录所调用，代码如下：

```
log_format main '$remote_addr - $remote_user [$time_local] "$request" '
            '$status $body_bytes_sent "$http_referer" '
            '"$http_user_agent" $request_time';
```

Nginx 日志格式内部变量及函数参数说明如下：

- $remote_addr：记录客户端 IP 地址。
- $server_name：虚拟主机名称。
- $http_x_forwarded_for：HTTP 请求端真实的 IP。
- $remote_user：记录客户端用户名称。
- $request：记录请求的 URL 和 HTTP 协议。
- $status：记录返回 HTTP 请求的状态。
- $uptream_status：upstream 的状态。
- $ssl_protocol：SSL 协议版本。

- $body_bytes_sent：发送给客户端的字节数，不包括响应头的大小。
- $bytes_sent：发送给客户端的总字节数。
- $connection_requests：当前通过一个连接获得的请求数量。
- $http_referer：记录从哪个页面链接访问过来的。
- $http_user_agent：记录客户端浏览器相关信息。
- $request_length：请求的长度，包括请求行、请求头和请求正文。
- $msec：日志写入时间。
- $request_time：请求处理时间，单位为 s，精度为 ms，Nginx 接受用户请求的第一个字节到发送完响应数据的时间，包括接收请求数据时间、程序响应时间、输出、响应数据时间。
- $upstream_response_time：应用程序响应时间，Nginx 向后端服务建立连接开始到接受完数据然后关闭连接为止的总时间。

通过 Nginx 日志，可以简单分析 Web 网站的运行状态、数据报表、IP、UV(unique visitor)、PV(page view)访问量等需求，以下为常用需求分析：

(1) 统计 Nginx 服务器独立 IP 数，代码如下：

```
awk '{print $1}' access.log |sort -r|uniq -c | wc -l
```

(2) 统计 Nginx 服务器总 PV 量，代码如下：

```
awk '{print $7}' access.log |wc -l
```

(3) 统计 Nginx 服务器 UV 统计，代码如下：

```
awk '{print $11}' access.log |sort -r|uniq -c |wc -l
```

(4) 分析 Nginx 访问日志截至目前为止访问量前 20 的 IP 列表，代码如下：

```
awk '{print $1}' access.log|sort |uniq -c |sort -nr |head -20
```

(5) 分析 Nginx 访问日志早上 9 点至中午 12 点的总请求量，代码如下：

```
sed -n "/2016:09:00/,/2016:12:00/"p access.log
awk '/2017:09:00/,/2017:12:00/' access.log|wc -l
```

(6) 分析 Nginx 访问日志总的独立 IP 数，代码如下：

```
awk '{print $1}' access.log |sort |uniq -c|wc -l
```

(7) 分析 Nginx 访问日志截至目前为止访问量前 20 的 IP 列表，代码如下：

```
awk '{print $1}' access.log|sort |uniq -c |sort -nr |head -20
```

(8) 分析 Nginx 访问日志截至目前为止访问量前 20 的 IP 列表，代码如下：

```
awk '{print $1}' access.log|sort |uniq -c |sort -nr |head -20
```

(9) 分析 Nginx 访问日志状态码 404、502、503、500、499 等错误信息页面，打印错误出现次数大于 20 的 IP 地址，代码如下：

```
awk '{if ( $9~/502|499|500|503|404/) print $1, $9}' access.log|sort|uniq -c|sort -nr |awk
'{if( $1>20) print $2}'
```

(10) 分析 Nginx 访问日志访问最多的页面，代码如下：

```
awk '{print $7}'   access.log |sort |uniq -c|sort -nr|head -20
```

(11) 分析 Nginx 访问日志请求处理时间大于 5s 的 URL，并打印出时间、URL、访客 IP，代码如下：

```
awk '{if ( $NF>5) print $NF, $7, $1}' access.log|sort -nr|more
```

14.13 Nginx 日志切割案例

Nginx Web 服务器每天会产生大量的访问日志，而且不会自动地进行切割，如果持续数天访问，将会导致该 access.log 日志文件容量非常大，不便于 SA 查看相关的网站异常日志。

可以基于 shell 脚本结合 crontab 计划任务对 Nginx 日志进行自动、快速的切割，其切割的方法使用 mv 命令即可，代码如下，详情如图 14-8 所示。

```
#!/bin/bash
#auto mv nginx log shell
#by author jfedu.net
S_LOG=/usr/local/nginx/logs/access.log
D_LOG=/data/backup/'date +%Y%m%d'
echo -e "\033[32mPlease wait start cut shell scripts...\033[1m"
sleep 2
if [ ! -d $D_LOG ]; then
        mkdir -p $D_LOG
fi
mv $S_LOG $D_LOG
kill -USR1 'cat /usr/local/nginx/logs/nginx.pid'
echo "-------------------------------------------"
echo "The Nginx log Cutting Successfully!"
echo "You can access backup nginx log $D_LOG/access.log files."
```

将以上脚本内容写入 auto_nginx_log.sh 文件，crontab /var/spool/cron/root 文件中添加如下代码，每天凌晨自动切割日志。

```
0  0  *   *   *   /bin/sh /data/sh/auto_nginx_log.sh >>/tmp/nginx_cut.log 2>&1
```

```
+ S_LOG=/usr/local/nginx/logs/access.log
++ date +%Y%m%d
+ D_LOG=/data/backup/20170525
+ echo -e '\033[32mPlease wait start cut shell scripts...\033[1m'
Please wait start cut shell scripts...
+ sleep 2
+ '[' '!' -d /data/backup/20170525 ']'
+ mv /usr/local/nginx/logs/access.log /data/backup/20170525
++ cat /usr/local/nginx/logs/nginx.pid
+ kill -USR1 48298
+ echo ------------------------------------------
------------------------------------------
+ echo 'The Nginx log Cutting Successfully!'
The Nginx log Cutting Successfully!
+ echo 'You can access backup nginx log /data/backup/20170525/access.log
You can access backup nginx log /data/backup/20170525/access.log files.
```

图 14-8　Nginx 日志切割

14.14　Nginx 防盗链配置案例

防盗链的含义是网站内容本身不在自己公司的服务器上，而通过技术手段，直接调用其他公司的服务器网站数据，而向最终用户提供此内容。一些小网站盗用高访问量网站的音乐、图片、软件的链接，然后放置在自己的网站中，通过这种方法盗取高访问量网站的空间和流量。

网站每天访问量很大，而且占用了很多不必要的带宽，浪费资源，所以必须采取一些限制措施。防盗链其实就是采用服务器端编程技术，通过 URL 过滤、主机名等实现的防止盗链的软件。

例如 http://www.jfedu.net/linux/页面，如果没有配置防盗链，别人就能轻而易举地在其网站上引用该页面。Nginx 防盗链配置代码如下：

```
server {
        listen        80;
        server_name jfedu.net www.jfedu.net;
        location / {
            root      html/b;
            index     index.html index.htm;
        }
        location ~ * \.(gif|jpg|png|swf|flv)${
                valid_referers none blocked jfedu.net * . jfedu.net;
                root html/b;
        if ( $invalid_referers) {
                # rewrite ^/ http://www.jfedu.net/403.html;
                return 403;
                }
        }
}
```

Nginx 防盗链参数详解如下：

- valid_referers：表示可用的 referers 设置。
- none：表示没有 referers，直接通过浏览器或者其他工具访问。
- blocked：表示有 referers，但是被代理服务器或者防火墙隐藏。
- jfedu.net：表示通过 jfedu.net 访问的 referers。
- *.jfedu.net：表示通过 *.jfedu.net 访问的 referers，* 表示任意 host 主机。

除了以上方法，按照如下方法设置也可以实现防盗链：

```
location ~ * \.(gif|jpg|png|swf|flv)$
if ( $host != '*.jfedu.net') {
    return 403;
}
```

防盗链测试，找另外一台测试服务器，基于 Nginx 发布如下 test.html 页面，代码如下，去调用 www.jfedu.net 官网的 test.png 图片，由于 www.jfedu.net 官网设置了防盗链，所以无法访问该图片。

```
<html>
<h1>TEST Nginx PNG</h1>
<img src="http://www.jfedu.net/test.png">
</html>
```

默认没有配置 Nginx 防盗链，网站正常调用 www.jfedu.net 的 logo 图片，访问如图 14-9 所示。

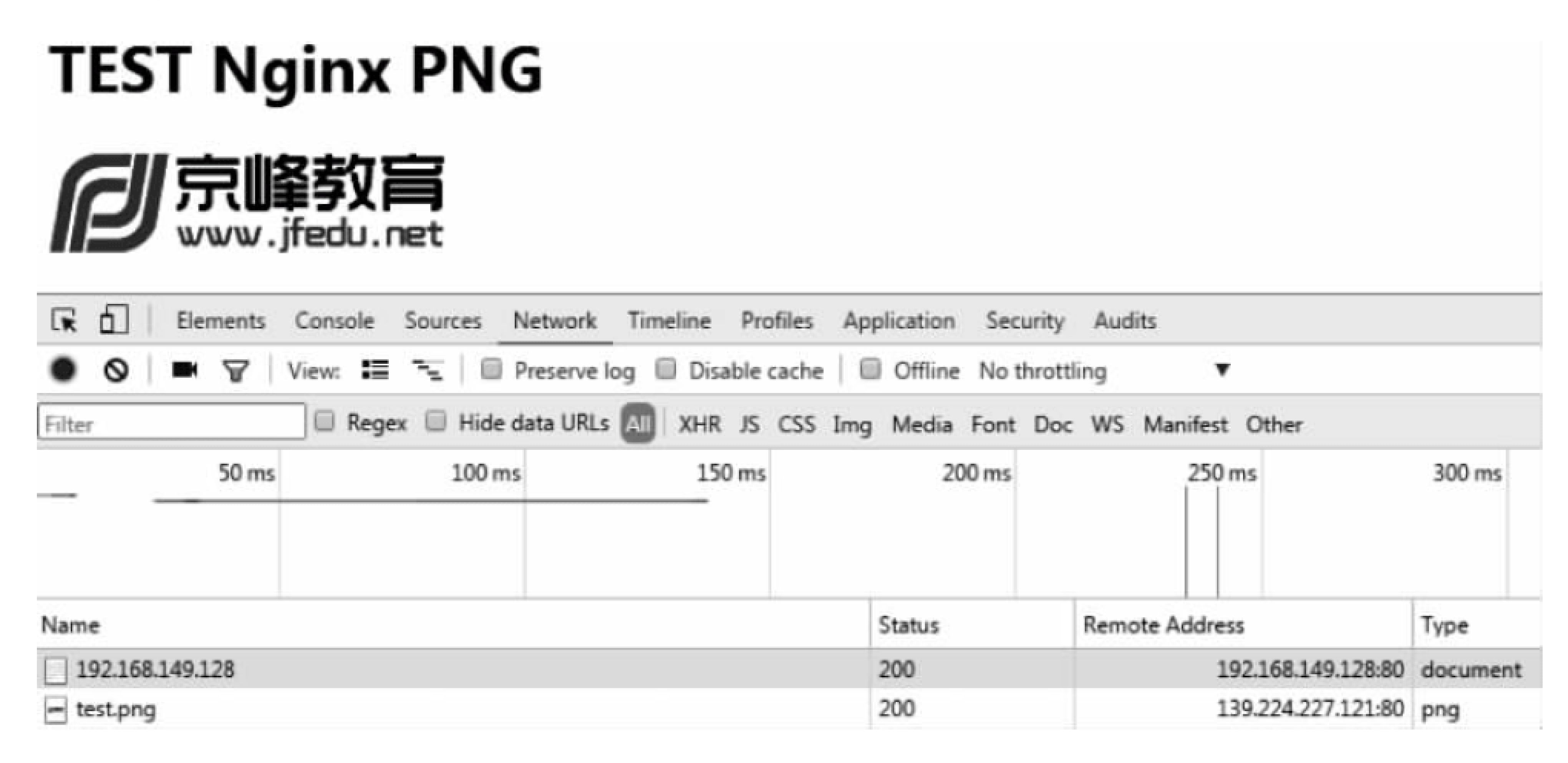


图 14-9　Nginx 无防盗链正常调用图片

配置 Nginx 防盗链，网站无法正常调用 www.jfedu.net 的 logo 图片，访问如图 14-10 所示。

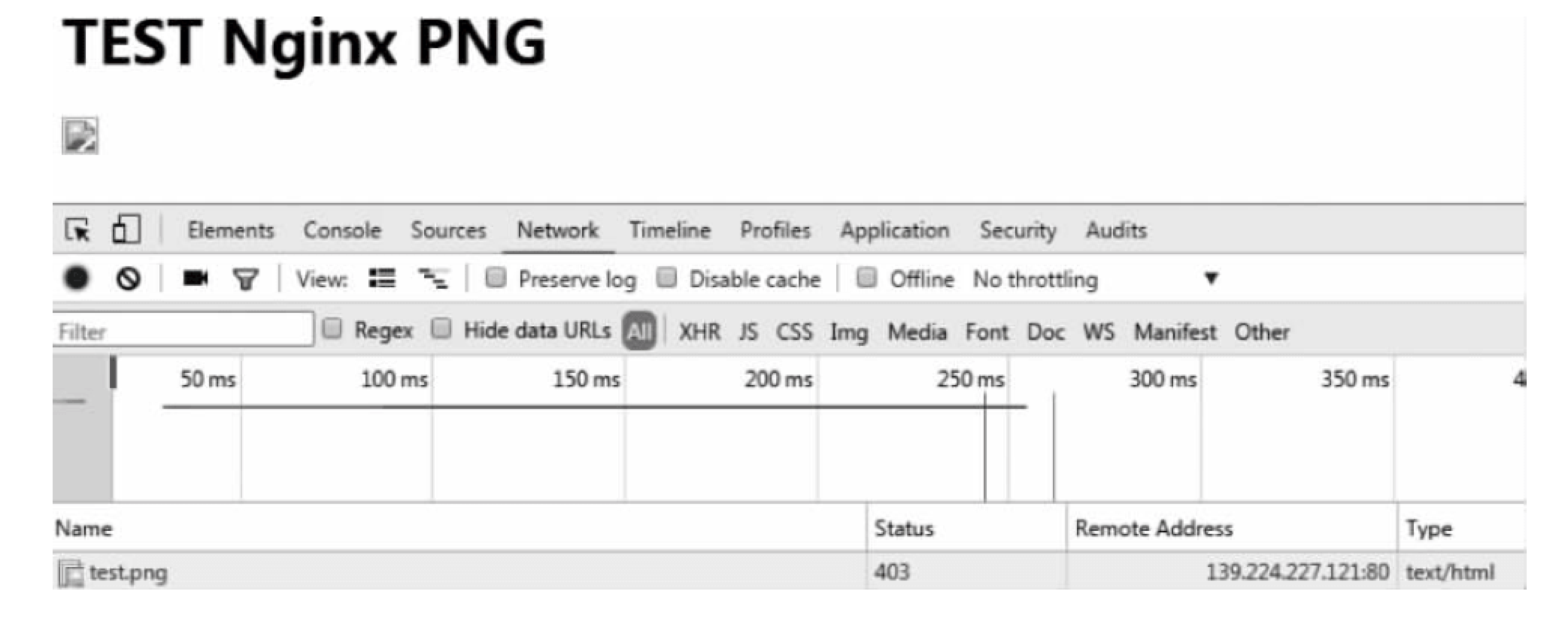


图 14-10 Nginx 防盗链 403 禁止访问

14.15 Nginx HTTPS 企业实战

超文本传输安全协议(hyper text transfer protocol over secure socket layer,HTTPS),是以安全为目标的 HTTP 通道,简单来说就是 HTTP 的安全版。HTTPS 由两部分组成,即 HTTP + SSL / TLS,在 HTTP 基础上又加了一层处理加密信息的模块,服务端和客户端的信息传输都会通过 TLS 进行加密,传输的数据都是加密后的数据。

为了解决 HTTP 协议的这一缺陷,需要使用另一种协议 HTTPS。为了数据传输的安全,HTTPS 在 HTTP 的基础上加入了 SSL 协议,SSL 依靠证书来验证服务器的身份,并为浏览器和服务器之间的通信加密。

SSL 证书是一种数字证书,它使用 secure socket layer 协议在浏览器和 Web 服务器之间建立一条安全通道,从而实现数据信息在客户端和服务器之间的加密传输,保证双方传递信息的安全性,不可被第三方窃听。而且用户可以通过服务器证书验证他所访问的网站是否真实可靠。

加密的 HTTP 传输通道,浏览器访问格式为 https://url,其基于 HTTP+TLS,现被广泛应用于互联网上安全敏感的通信,例如安全登录、订单交易、支付结算等方面。

HTTPS 和 HTTP 的区别在于 HTTP 协议被用于在 Web 浏览器和网站服务器之间传递信息,以明文方式发送内容,不提供任何方式的数据加密,如果攻击者截取了 Web 浏览器和网站服务器之间的传输报文,就可以直接读懂其中的信息。因此 HTTP 协议不适合传输一些敏感信息,比如信用卡号、密码等。

HTTPS 加密、解密、验证的完整过程如图 14-11 所示。

HTTPS 传输过程的 8 个步骤内容详解如下:

(1) 客户端发起 HTTPS 请求,用户在浏览器里输入 HTTPS 网址,然后连接到 Nginx server 的 443 端口。

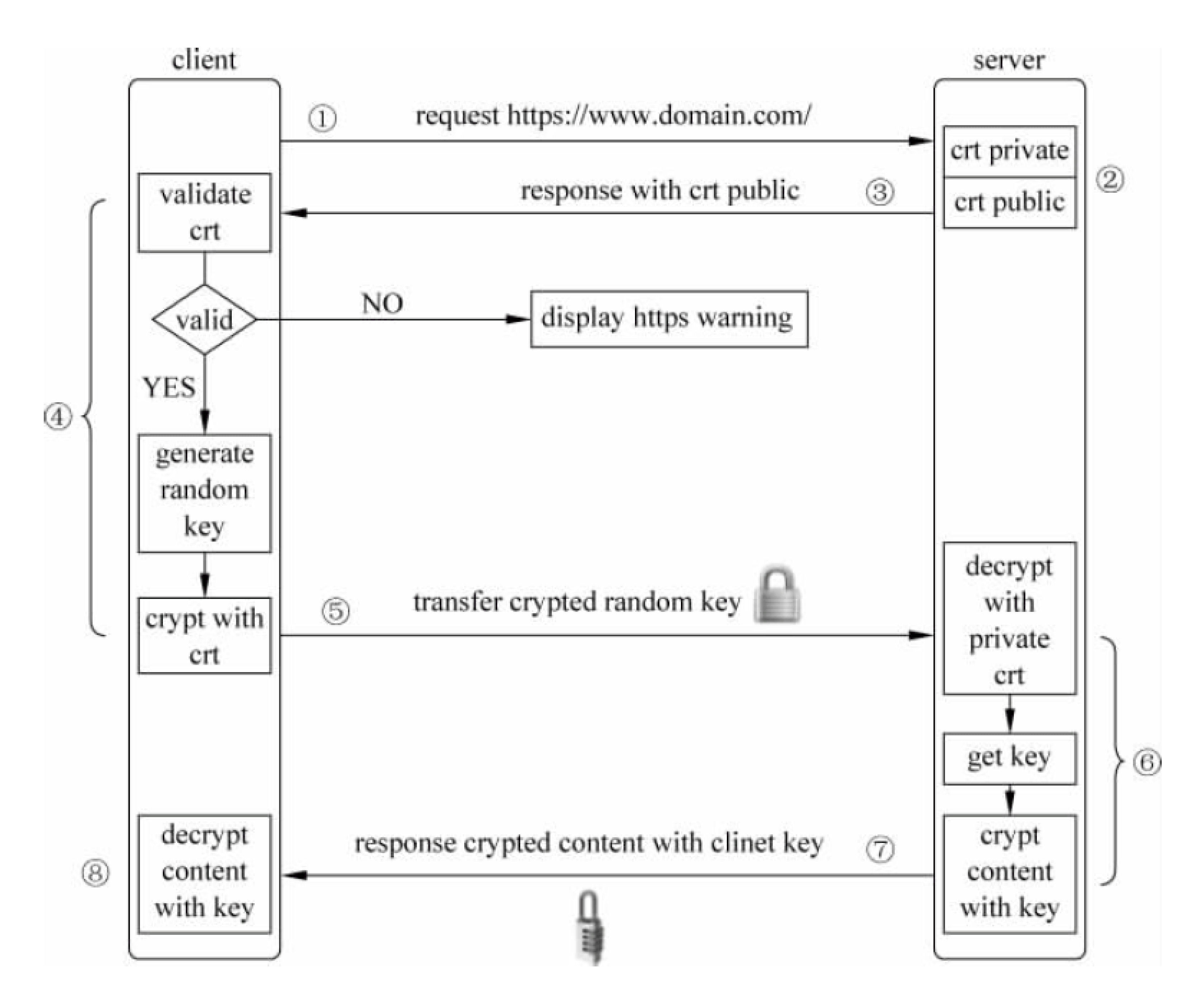


图 14-11　HTTPS 传输原理

(2) 服务器端采用的 HTTPS 协议有一套数字证书，该证书可以自行配置，也可以向证书管理组织去申请，该证书其本质是公钥和私钥。

(3) 将公钥传送证书传递给客户端，证书包含了很多信息，例如证书的颁发机构，过期时间、网址、公钥等。

(4) 客户端解析证书，由客户端的 TLS 来完成，首先会验证公钥是否有效，比如颁发机构，过期时间等，如果发现异常，则会弹出警告框，提示证书存在问题。如果证书没有问题，就会生成一个随机值，然后用证书对该随机值进行加密。

(5) 将证书加密后的随机值传送至服务器，让服务端获取该随机值，后续客户端和服务端的通信可以通过该随机值来进行加密解密。

(6) 服务端用私钥解密后，得到了客户端传过来的随机值，然后把内容通过该值进行对称加密。

(7) 服务器端将用私钥加密后的信息发给客户端。

(8) 客户端用之前生成的私钥来解密服务端发送过来的信息，获取解密后的内容。

HTTPS 证书申请与颁发方法如下：

(1) 生成 HTTPS 证书，可以使用 openssl 生成服务端 RSA 密钥及证书，生成的命令如下，详情如图 14-12 所示。

```
openssl genrsa -des3 -out server.key 1024
```

```
[root@node1 ~]# openssl genrsa -des3 -out server.key 1024
Generating RSA private key, 1024 bit long modulus
................++++++
............++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
139785688930120:error:28069065:lib(40):UI_set_result:result tc
ll:ui_lib.c:869:You must type in 4 to 8191 characters
Enter pass phrase for server.key:
139785688930120:error:28069065:lib(40):UI_set_result:result tc
```

图 14-12 openssl 生成 RSA 秘钥及证书

(2) 创建签名请求的证书(CSR),命令如下,详情如图 14-13 所示。

```
openssl req -new -key server.key -out server.csr
```

```
[root@node1 ~]# openssl req -new -key server.key -out server.csr
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:CN
State or Province Name (full name) []:BeiJing
Locality Name (eg, city) [Default City]:BeiJing
Organization Name (eg, company) [Default Company Ltd]:jfedu
Organizational Unit Name (eg, section) []:jfedu.net
Common Name (eg, your name or your server's hostname) []:jfedu.net
Email Address []:support@jfedu.net

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:111111
An optional company name []:jfedu
```

图 14-13 openssl 创建签名请求证书

(3) 加载 SSL 支持的 Nginx 并使用私钥时去除口令,命令如下,详情如图 14-14 所示。

```
cp server.key server.key.bak
openssl rsa -in server.key.bak -out server.key
```

```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:111111
An optional company name []:jfedu
[root@node1 ~]#
[root@node1 ~]# cp server.key server.key.bak
[root@node1 ~]# openssl rsa -in server.key.bak -out server.key
Enter pass phrase for server.key.bak:
writing RSA key
[root@node1 ~]#
```

图 14-14 openssl 去除口令

(4) 自动签发证书,命令如下,详情如图 14-15 所示。

```
openssl x509 -req -days 10240 -in server.csr -signkey server.key -out server.crt
```

HTTPS 证书生成完毕,配置 Nginx 以及证书整合,具体步骤如下:

(1) 安装 Nginx 并加入 SSL 模块支持,并将证书复制至 nginx.conf 目录,命令如下:

```
./configure --prefix=/usr/local/nginx --with-http_ssl_module &&make &&make install
cp server.crt server.key /usr/local/nginx/conf/
```

```
[root@node1 ~]# openssl x509 -req -days 10240 -in server.csr -signkey server.ke
t
Signature ok
subject=/C=CN/ST=BeiJing/L=BeiJing/O=jfedu/OU=jfedu.net/CN=jfedu.net/emailAddre
.net
Getting Private key
[root@node1 ~]#
```


图 14-15　openssl 自动签发证书

（2）nginx.conf 配置文件内容如下：

```
# HTTPS server
server {
        listen              443 ssl;
        server_name         www.jfedu.net localhost;
        ssl_certificate     server.crt;
        ssl_certificate_key server.key;
        ssl_session_cache   shared:SSL:1m;
        ssl_session_timeout 5m;
        ssl_ciphers HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers on;
        location / {
            root    html;
            index index.html index.htm;
        }
}
```

（3）最后重启 Nginx 服务，然后访问 https://www.jfedu.net 或者 https://ip/即可，如图 14-16 所示。

```
[root@node1 conf]# /usr/local/nginx/sbin/nginx  -t
nginx: the configuration file /usr/local/nginx/conf/nginx.conf s
nginx: configuration file /usr/local/nginx/conf/nginx.conf test
[root@node1 conf]#
[root@node1 conf]#
[root@node1 conf]# /usr/local/nginx/sbin/nginx
[root@node1 conf]#
[root@node1 conf]#
[root@node1 conf]# netstat -tnl |grep 443
tcp        0      0 0.0.0.0:443             0.0.0.0:*
[root@node1 conf]#
```

(a) Nginx监听443端口

https://192.168.1.12

Welcome to nginx!

If you see this page, the nginx web server is successfully inst working. Further configuration is required.

For online documentation and support please refer to nginx.c Commercial support is available at nginx.com.

Thank you for using nginx.

(b) 客户端访问Nginx HTTPS

图 14-16　重启 Nginx 服务、访问 HTTPS

第三篇 Linux 高级篇

Linux 进阶篇总共包含 9 个章节，第 15～25 章学习内容分别为 Linux 性能优化企业实战、大数据备份企业实战、shell 企业编程基础、shell 编程高级企业实战、自动化运维发展前景、Puppet 自动运维企业实战、Ansible 自动运维企业实战、Jenkins 持续集成企业实战、Linux 高可用集群实战、实战 Docker 虚拟化技术、Openstack+KVM 构建企业私有云。

读者通过对高级篇 11 个章节的深入学习，可以独立维护和管理企业上百台、甚至上千台服务器，能够在企业中独当一面，打造企业级千万 PV 门户网站架构。

同时读者能够掌握对 MySQL 2TB 大数量的备份，通过对 Linux 服务器内核进行优化、内核故障进行排错，服务器异常可以被快速解决，并通过编写企业生产环境中各种 shell 脚本工具，可以实现网站自动化维护和部署。shell 编程高级企业实战这一章讲述了 11 个高级实战脚本案例，满足企业在各种场景中的使用。可以基于 shell 开发各种脚本，如构建网站服务器数据备份、LAMP 及 LNMP 一键安装部署、服务器硬件信息收集存入 DB、MySQL 主从实战、自动修改千台服务器 IP、Zabbix 自动部署客户端、Nginx 及 Tomcat 自动部署、Docker 虚拟化管理平台、Bind 高级管理等脚本。

对 Linux 高级篇的学习能够使读者完全胜任万台服务器的维护和管理，基于 Puppet 各种案例实现主动部署管理、客户端自动获取配置、批量管理服务器等，通过轻量级 Ansible 自动化部署工具，可以实现至少 1000 台服务器的运维和管理，并通过各种资源模块对服务器进行管理，同时还可以编写 PlayBook 剧本实现对服务器流程化管理，减少人工干预，实现对服务器和 Web 网站高效维护。

高级篇引入 Jenkins 自动化部署平台，讲述了传统网站部署、主流网站部署的方法，基于 Jenkins 构建的企业级自动化平台，支持 SVN、GIT 仓库，并结合 Ansible 自动化运维工具可以打造企业级自动化部署平台，让运维工作更加轻松。

本篇第 23 章以 9 个企业级高级实战集群部署为例，如 Nginx＋keepalived、Redis＋keepalived、LVS＋keepalived、Haproxy＋keepalived 满足企业各个应用环境的部署。第 24 章介绍了 Docker 虚拟化技术；第 25 章介绍了 Openstack＋KVM 构建企业私有云技术，做到真正学以致用，满足企业需求。

第 15 章



Linux 性能优化企业实战

随着企业网站访问量越来越大，服务器的压力也逐渐增加，主要体现在 CPU 使用率、内存、硬盘、网卡流量等方面资源占用情况很高。此时需对服务器性能进行调优，尽量在保持服务器的现有数量下，然后对其各个环节参数进行优化。

本章向读者介绍 Linux 企业级性能服务器优化、TCP/IP 报文、TCP 三次握手及四次断开、Linux 内核深入优化、Linux 内核故障解决方案以及对 Linux 性能进行评估等内容。

15.1 TCP/IP 报文详解

TCP/IP 定义了电子设备如何连入因特网，以及数据如何在它们之间传输的标准。协议采用了 4 层的层级结构，每一层都呼叫它的下一层所提供的协议来完成自己的需求。

TCP 负责发现传输的问题，一有问题就发出信号，要求重新传输，直到所有数据安全正确地传输到目的地，而 IP 是给因特网的每台联网设备规定一个地址。TCP/IP 协议数据封装的过程包括用户数据经过应用层协议封装后传递给传输层，传输层封装 TCP 头部，交给网络层，网络层封装 IP 头部后，再交给数据链路层，数据链路层封装 Ethernet 帧头和帧尾，交给物理层，物理层以比特流的形式将数据发送到物理线路上。

不同的协议层对数据包有不同的称谓，数据包在传输层叫作段(segment)，在网络层叫作数据报(datagram)，在链路层叫作帧(frame)。数据封装成帧后发到传输介质上，到达目的主机后每层协议再剥掉相应的首部，最后将应用层数据交给应用程序处理，如图 15-1 所示。

优化 Linux 服务器，需要了解 TCP 协议相关信息，例如 TCP/IP 数据报文的内容是如何传输的，IP 数据包报文详细结构图如图 15-2 所示。

IP 数据包详解如下：

(1) source port 和 destination port：分别占用 16 位，表示源端口号和目的端口号，用于区别主机中的不同进程。而 IP 地址是用来区分不同的主机的，源端口号和目的端口号配合上 IP 首部中的源 IP 地址和目的 IP 地址就能唯一的确定一个 TCP 连接。

(2) sequence number：用来标识从 TCP 发端向 TCP 收端发送的数据字节流，它表示在

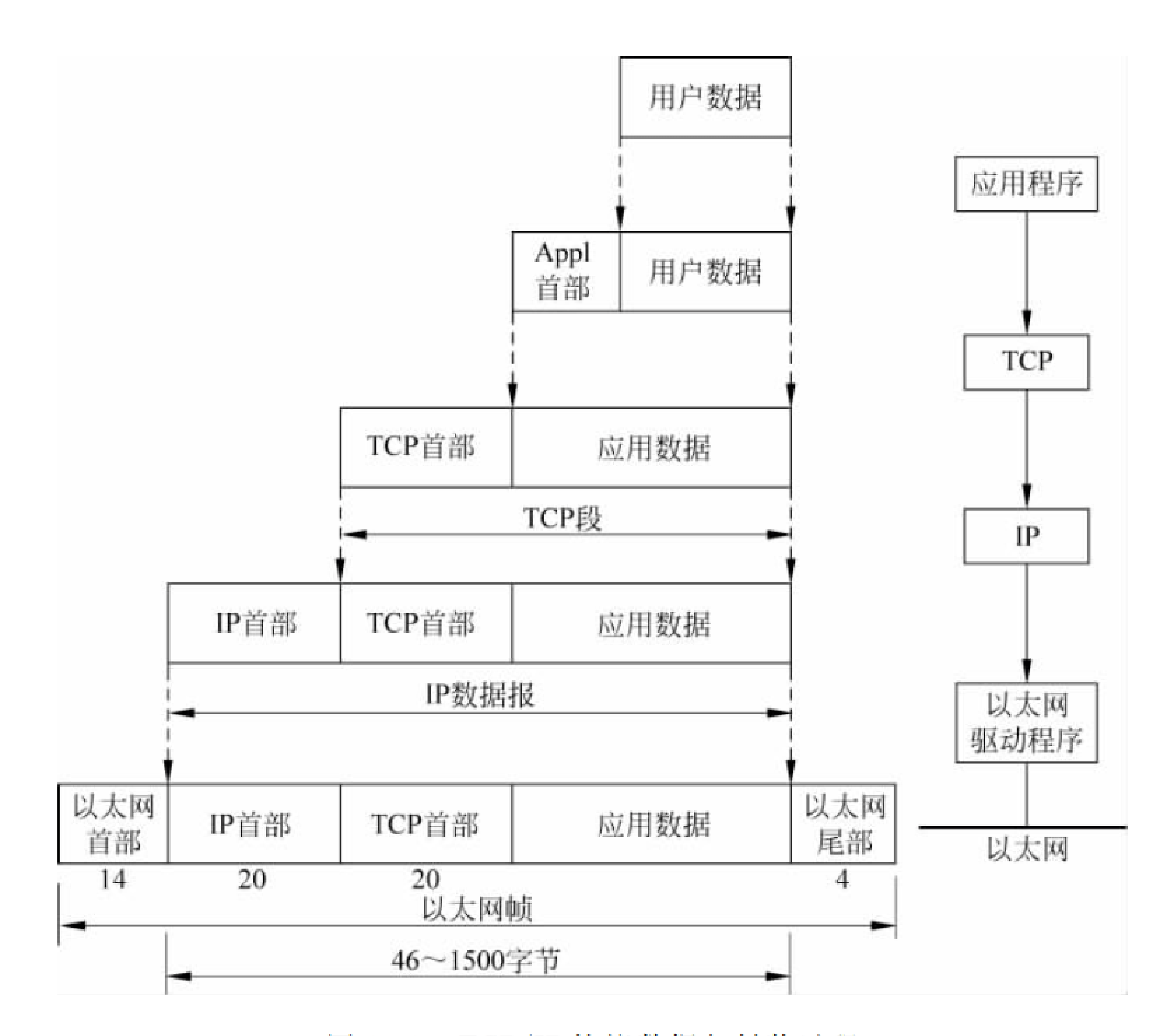


图 15-1　TCP/IP 协议数据包封装过程

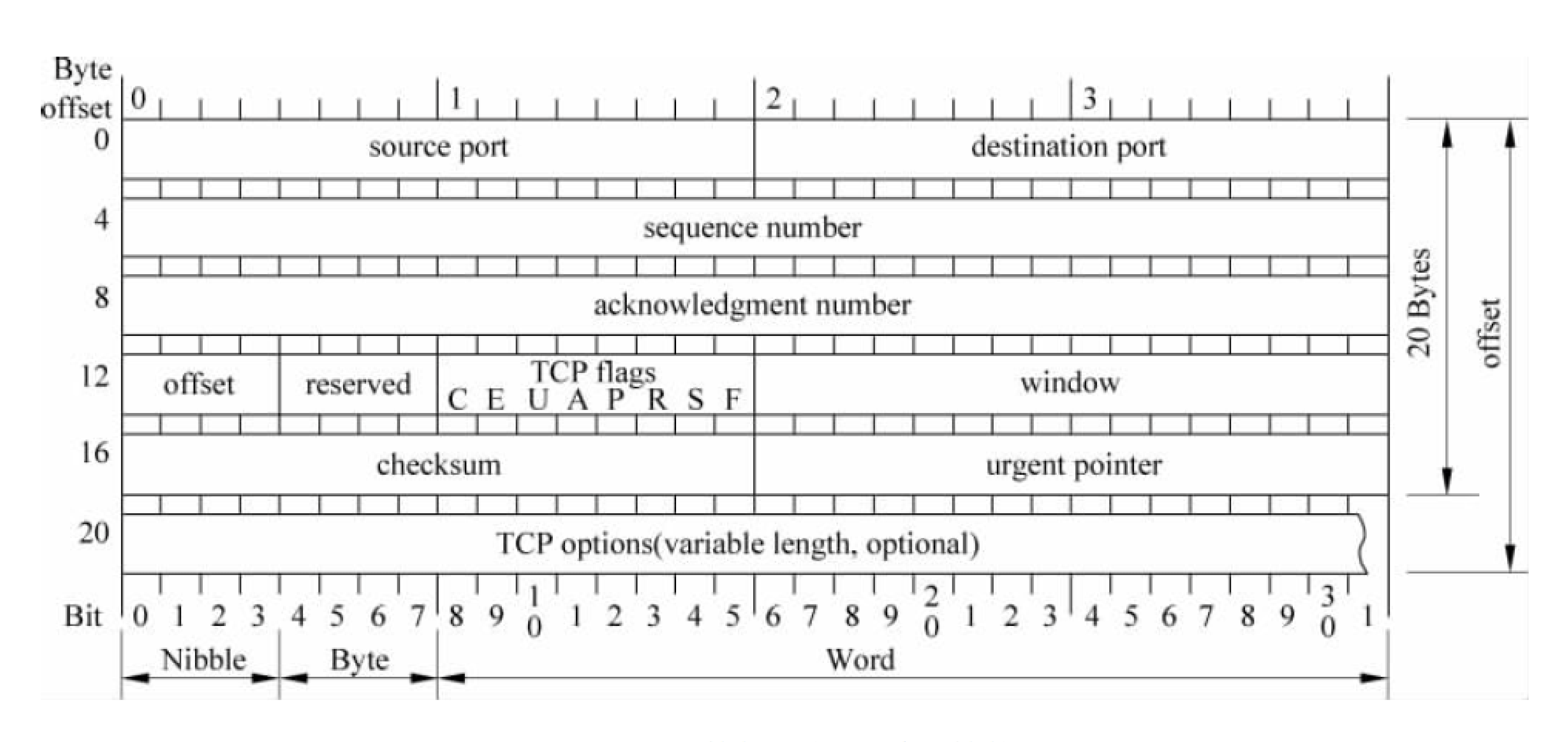


图 15-2　IP 数据包报文详细结构图

这个报文段中的第一个数据字节在数据流中的序号，主要用来解决网络报乱序的问题。

（3）acknowledgment number：32 位确认序列号包含发送确认的一端所期望收到的下一个序号，因此，确认序号应当是上次已成功收到数据字节序号加 1。不过，只有当标志位中的 ACK 标志（下面介绍）为 1 时该确认序列号的字段才有效。主要用来解决不丢包的

问题。

(4) offset：给出首部中32位的数目，需要这个值是因为任选字段的长度是可变的。这个字段占用4位（最多能表示15个32位的的字，即4×15=60个字节的首部长度），因此TCP最多有60字节的首部。然而，没有任选字段，正常的长度是20字节。

(5) TCP flags：TCP首部中有6个标志位，它们中的多个可同时被设置为1，主要是用于操控TCP的状态机，依次为URG，ACK，PSH，RST，SYN，FIN。每个标志位的含义如下：

- URG：此标志表示TCP包的紧急指针域（下面即将介绍）有效，用来保证TCP连接不被中断，并且督促中间层设备要尽快处理这些数据。
- ACK：此标志表示应答域有效，就是说前面所说的TCP应答号将会包含在TCP数据包中。有两个取值0和1，为1则表示应答域有效，反之为0。
- PSH：这个标志位表示push操作。所谓push操作就是指在数据包到达接收端以后，立即传送给应用程序，而不是在缓冲区中排队。
- RST：这个标志表示连接复位请求。用来复位那些产生错误的连接，也被用来拒绝错误和非法的数据包。
- SYN：表示同步序号，用来建立连接。SYN标志位和ACK标志位搭配使用，当连接请求的时候，SYN=1，ACK=0；连接被响应的时候，SYN=1，ACK=1。这个标志的数据包经常被用来进行端口扫描。扫描者发送一个只有SYN的数据包，如果对方主机响应了一个数据包回来，就表明这台主机存在这个端口。但是由于这种扫描方式只是进行TCP三次握手的第一次握手，因此这种扫描的成功表示被扫描的机器不是很安全，一台安全的主机将会强制要求一个连接严格地进行TCP的三次握手。
- FIN：表示发送端已经达到数据末尾，也就是说双方的数据传送完成，没有数据可以传送了，发送FIN标志位的TCP数据包后，连接将被断开。这个标志的数据包也经常被用于进行端口扫描。

(6) window：窗口大小，也就是有名的滑动窗口，用来进行流量控制。

15.2 TCP三次握手及四次断开

TCP是面向连接的，无论哪一方向在另一方发送数据之前，都必须先在双方之间建立一条连接。在TCP/IP协议中，TCP协议提供可靠的连接服务，连接是通过三次握手进行初始化的。三次握手的目的是同步连接双方的序列号和确认号并交换TCP窗口大小信息，如图15-3所示。

(1) TCP三次握手原理如下：

- 第一次握手：建立连接。客户端发送连接请求报文段，将SYN位置为1，sequence number为x，然后客户端进入SYN_SEND状态，等待服务器的确认。

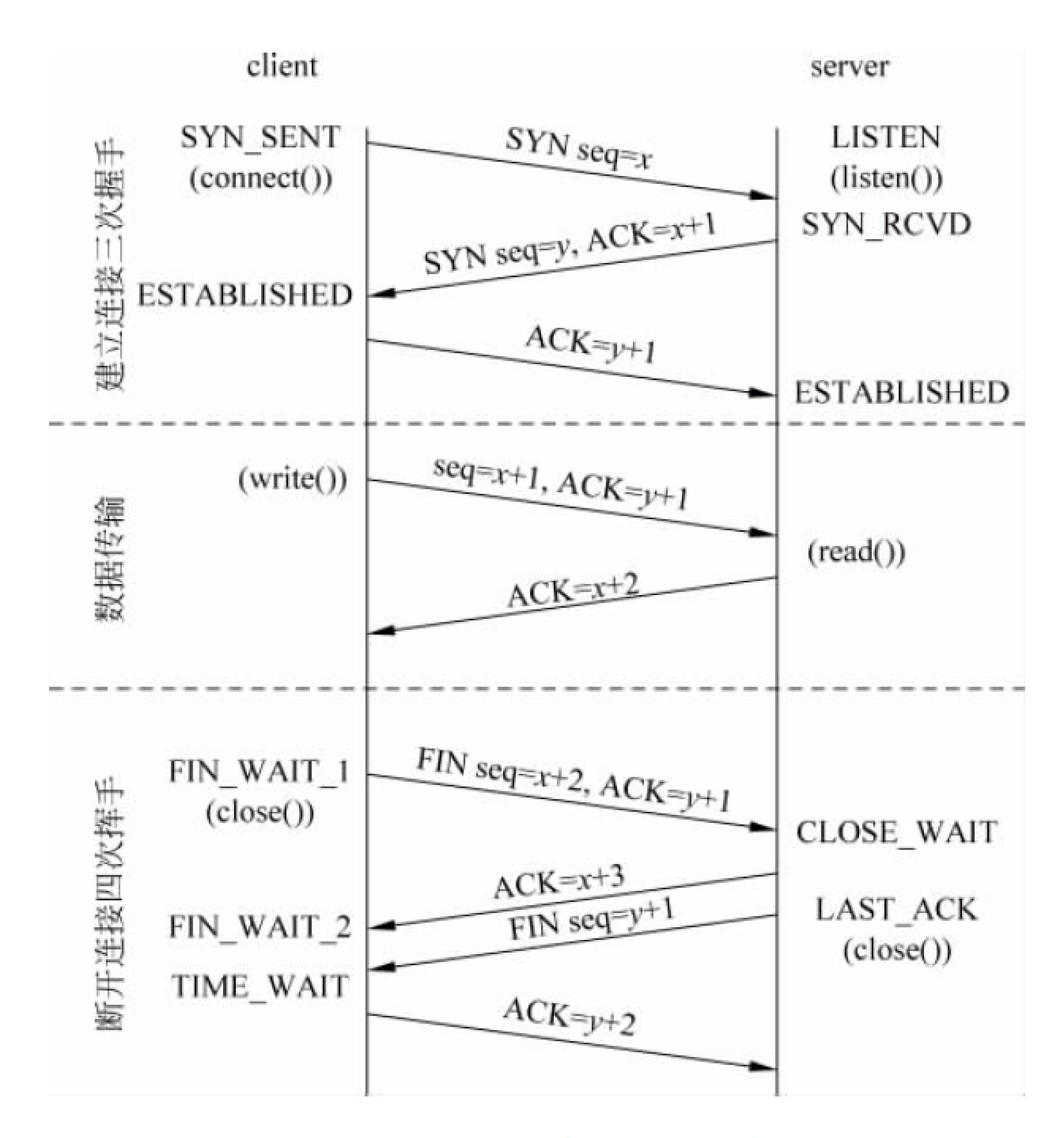


图 15-3　TCP 三次握手及四次断开

- 第二次握手：服务器收到 SYN 报文段。服务器收到客户端的 SYN 报文段，需要对这个 SYN 报文段进行确认，设置 acknowledgment number 为 $x+1$（sequence number+1）。同时，自己还要发送 SYN 请求信息，将 SYN 位置为 1，sequence number 为 y。服务器端将上述所有信息放到一个报文段（即 SYN+ACK 报文段）中，一并发送给客户端，此时服务器进入 SYN_RECV 状态。
- 第三次握手：客户端收到服务器的 SYN+ACK 报文段。然后将 acknowledgment number 设置为 $y+1$，向服务器发送 ACK 报文段，这个报文段发送完毕以后，客户端和服务器端都进入 ESTABLISHED 状态，完成 TCP 三次握手。

如图 15-4 所示为基于 tcpdump 抓取 TCP/IP 三次握手及数据包传输过程。

（2）TCP 四次挥手原理如下：

- 第一次挥手：主机 A（可以使客户端，可以是服务器端），设置 sequence number 和 acknowledgment number，向主机 B 发送一个 FIN 报文段。此时，主机 A 进入 FIN_WAIT_1 状态，这表示主机 A 没有数据要发送给主机 B。
- 第二次挥手：主机 B 收到了主机 A 发送的 FIN 报文段，向主机 A 回一个 ACK 报文段，acknowledgment number 为 sequence number 加 1，主机 A 进入 FIN_WAIT_2 状态。主机 B 告诉主机 A，我“同意”你的关闭请求。
- 第三次挥手：主机 B 向主机 A 发送 FIN 报文段，请求关闭连接，同时主机 B 进入

```
tcpdump -nn port 80 and host 192.168.149.129
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
10:25:02.979356 IP 192.168.149.129.60980 > 192.168.149.128.80: Flags [S], seq 1287430895, win 14600, options [mss 1460,sa
ckOK,TS val 507292290 ecr 0,nop,wscale 5], length 0
10:25:02.979395 IP 192.168.149.128.80 > 192.168.149.129.60980: Flags [S.], seq 983948645, ack 1287430896, win 14480, opti
ons [mss 1460,sackOK,TS val 558399769 ecr 507292290,nop,wscale 5], length 0
10:25:02.979583 IP 192.168.149.129.60980 > 192.168.149.128.80: Flags [.], ack 1, win 457, options [nop,nop,TS val 5072922
90 ecr 558399769], length 0
10:25:02.979744 IP 192.168.149.129.60980 > 192.168.149.128.80: Flags [P.], seq 1:124, ack 1, win 457, options [nop,nop,TS
 val 507292290 ecr 558399769], length 123
10:25:02.979791 IP 192.168.149.128.80 > 192.168.149.129.60980: Flags [.], ack 124, win 453, options [nop,nop,TS val 55839
9770 ecr 507292290], length 0
10:25:02.986587 IP 192.168.149.128.80 > 192.168.149.129.60980: Flags [P.], seq 1:269, ack 124, win 453, options [nop,nop,
TS val 558399776 ecr 507292290], length 268
10:25:02.986937 IP 192.168.149.129.60980 > 192.168.149.128.80: Flags [.], ack 269, win 490, options [nop,nop,TS val 50729
2297 ecr 558399776], length 0
10:25:02.987124 IP 192.168.149.128.80 > 192.168.149.129.60980: Flags [F.], seq 269, ack 124, win 453, options [nop,nop,TS
 val 558399777 ecr 507292297], length 0
10:25:02.987612 IP 192.168.149.129.60980 > 192.168.149.128.80: Flags [F.], seq 124, ack 270, win 490, options [nop,nop,TS
 val 507292298 ecr 558399777], length 0
10:25:02.987631 IP 192.168.149.128.80 > 192.168.149.129.60980: Flags [.], ack 125, win 453, options [nop,nop,TS val 55839
9778 ecr 507292298], length 0
^C
```

图 15-4　TCP 三次握手抓包分析

LAST_ACK 状态。

- 第四次挥手：主机 A 收到主机 B 发送的 FIN 报文段，向主机 B 发送 ACK 报文段，然后主机 A 进入 TIME_WAIT 状态。主机 B 收到主机 A 的 ACK 报文段以后，就关闭连接。此时，主机 A 等待 2MSL 后依然没有收到回复，则证明 server 端已正常关闭，这样，主机 A 也可以关闭连接。

如图 15-5 所示为基于 tcpdump 抓取 TCP/IP 四次挥手及数据包传输过程。

```
 val 507749963 ecr 558857442], length 123
10:32:40.652808 IP 192.168.149.128.80 > 192.168.149.129.60986: Flags [.], ack 124, win 453, options [nop,nop,TS val 55885
7443 ecr 507749963], length 0
10:32:40.655188 IP 192.168.149.128.80 > 192.168.149.129.60986: Flags [P.], seq 1:272, ack 124, win 453, options [nop,nop,
TS val 558857445 ecr 507749963], length 271
10:32:40.655767 IP 192.168.149.128.80 > 192.168.149.129.60986: Flags [F.], seq 272, ack 124, win 453, options [nop,nop,TS
 val 558857446 ecr 507749963], length 0
10:32:40.655877 IP 192.168.149.129.60986 > 192.168.149.128.80: Flags [.], ack 272, win 490, options [nop,nop,TS val 50774
9966 ecr 558857445], length 0
10:32:40.656597 IP 192.168.149.129.60986 > 192.168.149.128.80: Flags [F.], seq 124, ack 273, win 490, options [nop,nop,TS
 val 507749967 ecr 558857446], length 0
10:32:40.656616 IP 192.168.149.128.80 > 192.168.149.129.60986: Flags [.], ack 125, win 453, options [nop,nop,TS val 55885
7447 ecr 507749967], length 0
^C
10 packets captured
```

图 15-5　TCP 四次挥手抓包分析

15.3　优化 Linux 文件打开最大数

为了防止失控的进程破坏系统的性能，UNIX 和 Linux 会跟踪进程使用的大部分资源，并允许用户和系统管理员使用对进程的资源限制，例如控制某个进程打开的系统文件数、对某个用户打开系统进程数进行限制等，一般限制手段包括软限制和硬限制。具体说明如下：

- 软限制(soft limit)：内核实际执行的限制，任何进程都可以将软限制设置为任意小于或等于对进程限制的硬限制的值、最大线程数(noproc)和文件数(nofile)。
- 硬限制(hard limit)：可以在任何时候任何进程中设置，但硬限制只能由超级用户修改。

Linux 系统一切皆文件，对 Linux 进行各种操作，实际就是对文件进行操作，文件可分为普通文件、目录文件、链接文件和设备文件。而文件描述符(file descriptor)是内核为了高

效管理已被打开的文件所创建的索引，其值是一个非负整数(通常是小整数)，用于指代被打开的文件，所有执行I/O操作的系统调用都通过文件描述符。

Linux系统默认已经打开的文件描述符包括STDIN_FILENO 0表示标准输入、STDOUT_FILENO 1表示标准输出、STDERR_FILENO 2表示标准错误输出，默认打开一个新文件，它的文件描述符为3。

每个文件描述符与一个打开文件相对应，不同的文件描述符可以指向同一个文件。相同的文件可以被不同的进程打开，也可以在同一个进程中被多次打开。

Linux系统为每个进程维护了一个文件描述符表，该表的值都是从0开始的，在不同的进程中用户会看到相同的文件描述符，相同文件描述符有可能指向同一个文件，也有可能指向不同的文件。Linux内核对文件操作，维护了3个数据结构概念。

- 进程级的文件描述符表；
- 系统级的打开文件描述符表；
- 文件系统的i-node表。

其中进程级的描述符表的每一个条目记录了单个文件描述符的相关信息，例如控制文件描述符操作的一组标志及对打开文件句柄的引用。Linux内核对所有打开的文件都维护了一个系统级的描述符表(open file description table)。将描述符表中的记录行称为打开文件句柄(open file handle)，一个打开文件句柄存储了与一个打开文件相关的全部信息，详细信息如下：

- 当前文件偏移量；
- 打开文件时所使用的状态标识；
- 文件访问模式；
- 与信号驱动相关的设置；
- 对该文件i-node对象的引用；
- 文件类型和访问权限；
- 指针指向该文件所持有的锁列表；
- 文件的各种属性。

默认Linux内核对每个用户设置了打开文件最大数为1024，对于高并发网站，是远远不够的，需要将默认值调整到更大，调整方法如下：

- Linux每个用户打开文件最大数临时设置方法，重启服务器该参数无效，命令行终端执行如下命令：

```
ulimit -n 65535
```

- Linux每个用户打开文件最大数永久设置方法，将如下代码加入内核限制文件/etc/security/limits.conf的末尾：

```
*    soft    noproc      65535
*    hard    noproc      65535
```

```
*   soft    nofile      65535
*   hard    nofile      65535
```

上述设置为对每个用户分别设置 nofile、noproc 最大数，如果需要对 Linux 整个系统设置文件最大数限制，需要修改/proc/sys/fs/file-max 中的值，该值为 Linux 总文件打开数，例如设置为 echo 3865161233 >/proc/sys/fs/file-max。

15.4 内核参数的优化

Linux /proc/sys 目录下存放着多数内核的参数，并且可以在系统运行时进行更改，一般重新启动机器就会失效。而/etc/sysctl.conf 是一个允许改变正在运行中的 Linux 系统的接口，它包含一些 TCP/IP 堆栈和虚拟内存系统的高级选项，修改内核参数永久生效。

/proc/sys 下内核文件与配置文件 sysctl.conf 中变量存在着对应关系，即修改 sysct.conf 配置文件，其实是修改/proc/sys 相关参数，所以对 Linux 内核优化只需修改/etc/sysctl.conf 文件即可。以下为 BAT 企业生产环境/etc/sysct.conf 内核参数：

```
net.ipv4.ip_forward = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
kernel.sysrq = 0
kernel.core_uses_pid = 1
net.ipv4.tcp_syncookies = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.shmmax = 68719476736
kernel.shmall = 4294967296
net.ipv4.tcp_max_tw_buckets = 10000
net.ipv4.tcp_sack = 1
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_rmem = 4096 87380 4194304
net.ipv4.tcp_wmem = 4096 16384 4194304
net.core.wmem_default = 8388608
net.core.rmem_default = 8388608
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.core.netdev_max_backlog = 262144
net.core.somaxconn = 262144
net.ipv4.tcp_max_orphans = 3276800
net.ipv4.tcp_max_syn_backlog = 262144
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_synack_retries = 1
net.ipv4.tcp_syn_retries = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_mem = 94500000 915000000 927000000
net.ipv4.tcp_fin_timeout = 1
```

```
net.ipv4.tcp_keepalive_time = 30
net.ipv4.ip_local_port_range = 1024 65535
```

Linux 内核常见参数详解如下：

- net.ipv4.tcp_timestamps = 1：该参数控制 RFC 1323 时间戳与窗口缩放选项。
- net.ipv4.tcp_sack = 1：选择性应答(SACK)是 TCP 的一项可选特性，可以提高某些网络中所有可用带宽的使用效率。
- net.ipv4.tcp_fack = 1：打开 FACK(forward ACK)拥塞避免和快速重传功能。
- net.ipv4.tcp_retrans_collapse = 1：打开重传重组包功能，为 0 的时候关闭重传重组包功能。
- net.ipv4.tcp_syn_retries = 5：对于一个新建连接，内核要发送多少个 SYN 连接请求才决定放弃。
- net.ipv4.tcp_synack_retries = 5：tcp_synack_retries 显示或设定 Linux 在回应 SYN 要求时尝试多少次重新发送初始 SYN，ACK 封包后才决定放弃。
- net.ipv4.tcp_max_orphans = 131072：系统所能处理不属于任何进程的 TCP sockets 最大数量。
- net.ipv4.tcp_max_tw_buckets = 5000：系统同时保持 TIME_WAIT 套接字的最大数量，如果超过这个数字，TIME_WAIT 套接字将立刻被清除并打印警告信息，默认为 180000，设为较小数值此项参数可以控制 TIME_WAIT 套接字的最大数量，避免服务器被大量的 TIME_WAIT 套接字拖死。
- net.ipv4.tcp_keepalive_time = 30
  net.ipv4.tcp_keepalive_probes = 3
  net.ipv4.tcp_keepalive_intvl = 3

如果某个 TCP 连接在空闲 30s 后，内核才发起 probe(探查)，若 probe 3 次(每次 3s)即 tcp_keepalive_intvl 值不成功，内核才彻底放弃，认为该连接已失效。

- net.ipv4.tcp_retries1 = 3：放弃回应一个 TCP 连接请求前，需要进行多少次重试。
- net.ipv4.tcp_retries2 = 15：在丢弃激活(已建立通信状况)的 TCP 连接之前，需要进行多少次重试。
- net.ipv4.tcp_fin_timeout = 30：表示如果套接字由本端要求关闭，这个参数决定了它保持在 FIN-WAIT-2 状态的时间。
- net.ipv4.tcp_tw_recycle = 1：表示开启 TCP 连接中 TIME_WAIT sockets 的快速回收，默认为 0，表示关闭。
- net.ipv4.tcp_max_syn_backlog = 8192：表示 SYN 队列的长度，默认为 1024，加大队列长度为 8192，可以容纳更多等待连接的网络连接数。
- net.ipv4.tcp_syncookies = 1：TCP 建立连接的 3 路握手过程中，当服务端收到最初的 SYN 请求时，会检查应用程序的 syn_backlog 队列是否已满。启用 syncookie，

可以解决超高并发时的“can't connect”问题，但是会导致 TIME_WAIT 状态 fallback 为保持 2MSL 时间，高峰期时会导致客户端无可复用连接而无法连接服务器。

- net.ipv4.tcp_orphan_retries = 0：关闭 TCP 连接之前重试多少次。
- net.ipv4.tcp_mem = 178368　237824　356736：net.ipv4.tcp_mem[0]表示低于此值，TCP 没有内存压力，net.ipv4.tcp_mem[1]表示在此值下，进入内存压力阶段，net.ipv4.tcp_mem[2]表示高于此值，TCP 拒绝分配 socket。
- net.ipv4.tcp_tw_reuse = 1：表示开启重用，允许将 TIME_WAIT sockets 重新用于新的 TCP 连接。
- net.ipv4.ip_local_port_range = 1024 65000：表示用于向外连接的端口范围。
- net.ipv4.ip_conntrack_max = 655360：在内核内存中 netfilter 可以同时处理的“任务”(连接跟踪条目)。
- net.ipv4.icmp_ignore_bogus_error_responses = 1：开启恶意 icmp 错误消息保护。
- net.ipv4.tcp_syncookies = 1：开启 SYN 洪水攻击保护。

15.5　Linux 内核报错剖析

企业生产环境 Linux 服务器正常运行，由于某种原因会导致内核报错或者抛出很多信息，根据系统 SA 可以快速定位 Linux 服务器故障，Linux 内核日志一般存在 messages 日志中，可以通过命令 tail -fn 100 /var/log/messages 查看，以下为 Linux 内核常见报错日志及生产环境解决报错的方案。

(1) Linux 内核抛出 net.ipv4.tcp_max_tw_buckets 错误，代码如下：

```
Sep 23 04:45:55 localhost kernel: TCP: time wait bucket table overflow
Sep 23 04:45:55 localhost kernel: TCP: time wait bucket table overflow
Sep 23 04:45:55 localhost kernel: TCP: time wait bucket table overflow
Sep 23 04:45:55 localhost kernel: TCP: time wait bucket table overflow
Sep 23 04:45:55 localhost kernel: TCP: time wait bucket table overflow
Sep 23 04:45:55 localhost kernel: TCP: time wait bucket table overflow
Sep 23 04:45:55 localhost kernel: TCP: time wait bucket table overflow
Sep 23 04:45:55 localhost kernel: TCP: time wait bucket table overflow
Sep 23 04:45:55 localhost kernel: TCP: time wait bucket table overflow
Sep 23 04:45:55 localhost kernel: TCP: time wait bucket table overflow
```

根据 TCP 协议定义的三次握手及四次断开连接规定，发起 socket 主动关闭的一方 socket 将进入 TIME_WAIT 状态，TIME_WAIT 状态将持续 2 个 MSL(max segment lifetime)。如果该值设置过小导致，当系统 TIME_WAIT 数量超过默认设置的值时，即会抛出上述的警告信息，这时需要增加 net.ipv4.tcp_max_tw_buckets 的值，警告信息才会消除。当然这个值也不能设置过大，对于一个处理大量短连接的服务器，如果是由服务器主动

关闭客户端的连接，将导致服务器端存在大量的处于 TIME_WAIT 状态的 socket，甚至比处于 established 状态下的 socket 多得多，严重影响服务器的处理能力，甚至耗尽可用的 socket 而停止服务，TIME_WAIT 是 TCP 协议用以保证被重新分配的 socket 不会受到之前残留的延迟重发报文影响的机制，是 TCP 传输必要的逻辑保证。

（2）Linux 内核抛出 Too many open files 错误，代码如下：

```
Benchmarking localhost (be patient)
socket: Too many open files (24)
socket: Too many open files (24)
socket: Too many open files (24)
socket: Too many open files (24)
socket: Too many open files (24)
```

每个文件描述符与一个打开文件相对应，不同的文件描述符可以指向同一个文件。相同的文件可以被不同的进程打开，也可以在同一个进程中被多次打开。Linux 内核对应每个用户打开的文件最大数一般为 1024，需要将该值调高满足大并发网站的访问。

Linux 每个用户打开文件最大数永久设置方法，将以下代码加入内核限制文件/etc/security/limits.conf 的末尾，退出终端，重新登录即生效，代码如下：

```
*    soft    noproc     65535
*    hard    noproc     65535
*    soft    nofile     65535
*    hard    nofile     65535
```

（3）Linux 内核抛出 possible SYN flooding on port 80. Sending cookies 错误，代码如下：

```
May 31 14:20:14 localhost kernel: possible SYN flooding on port 80. Sending cookies.
May 31 14:21:28 localhost kernel: possible SYN flooding on port 80. Sending cookies.
May 31 14:22:44 localhost kernel: possible SYN flooding on port 80. Sending cookies.
May 31 14:25:33 localhost kernel: possible SYN flooding on port 80. Sending cookies.
May 31 14:27:06 localhost kernel: possible SYN flooding on port 80. Sending cookies.
May 31 14:28:44 localhost kernel: possible SYN flooding on port 80. Sending cookies.
May 31 14:28:51 localhost kernel: possible SYN flooding on port 80. Sending cookies.
May 31 14:31:01 localhost kernel: possible SYN flooding on port 80. Sending cookies.
```

此问题是由于 SYN 队列已满，而触发 SYN cookies，一般是由于大量的访问或者恶意访问导致，也称之为 SYN flooding 洪水攻击。

完整的 TCP 连接的三次握手，假设一个用户 A 向服务器发送了 SYN 报文后突然死机或掉线，那么服务器在发出 SYN+ACK 应答报文后是无法收到客户端的 ACK 报文的（第三次握手无法完成），这种情况下服务器端一般会重试（再次发送 SYN+ACK 给客户端）并等待一段时间后丢弃这个未完成的连接，这段时间的长度称为 SYN timeout，一般来说这个时间是分钟的数量级（大约为 30s～2min）。

一个用户出现异常导致服务器的一个线程等待 1min 并不是什么很大的问题，但如果

有恶意的攻击者大量模拟这种情况，服务器端将为了维护一个非常大的半连接列表而消耗非常多的资源，数以万计的半连接，即使是简单的保存并遍历也会消耗非常多的 CPU 时间和内存，何况还要不断对这个列表中的 IP 进行 SYN+ACK 的重试。

实际上如果服务器的 TCP/IP 栈不够强大，最后的结果往往是堆栈溢出崩溃，即使服务器端的系统足够强大，服务器端也将忙于处理攻击者伪造的 TCP 连接请求而无暇理睬客户的正常请求（毕竟客户端的正常请求比率非常之小），此时从正常客户的角度看来，服务器失去响应，服务器拒绝提供服务，服务器受到了 DDOS 攻击，这里攻击的手段为 DDOS 中 SYN flooding 攻击（SYN 洪水攻击）。

防护 DDOS 攻击有两种手段：一是基于硬件专业防火墙；二是基于 Linux 内核简单防护。如果攻击流量特别大，单纯配置内核参数是无法抵挡的，还得依靠专业级硬件防火墙，以下为 Linux 内核防护 DDOS 优化参数，添加如下代码即可：

```
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_keepalive_time = 1200
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.ip_local_port_range = 1024 65000
net.ipv4.tcp_max_syn_backlog = 8192
net.ipv4.tcp_max_tw_buckets = 8000
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_syn_retries = 2
```

（4）Linux 内核抛出 nf_conntrack：table full，dropping packet. 错误，代码如下：

```
May  6 11:15:07 localhost kernel: nf_conntrack:table full, dropping packet.
May  6 11:19:13 localhost kernel: nf_conntrack:table full, dropping packet.
May  6 11:20:34 localhost kernel: nf_conntrack:table full, dropping packet.
May  6 11:23:12 localhost kernel: nf_conntrack:table full, dropping packet.
May  6 11:24:07 localhost kernel: nf_conntrack:table full, dropping packet.
May  6 11:24:13 localhost kernel: nf_conntrack:table full, dropping packet.
May  6 11:25:11 localhost kernel: nf_conntrack:table full, dropping packet.
May  6 11:26:25 localhost kernel: nf_conntrack:table full, dropping packet.
```

由于该服务器开启了 iptables 防火墙，Web 服务器收到了大量的连接，iptables 会把所有的连接都做连接跟踪处理，这样 iptables 就会有一个连接跟踪表，当这个表满的时候，就会出现上面的错误。ip_conntrack 是 Linux NAT 的一个跟踪连接条目的模块，ip_conntrack 模块会使用一个哈希表记录 TCP 通信协议的 established connection 记录。

如果是 CentOS 6.X 系统，需执行 modprobe nf_conntrack 命令，然后在内核优化文件中加入如下代码，sysctl -p 使其内核文件生效，即可解决该报错。

```
net.nf_conntrack_max = 655360
net.netfilter.nf_conntrack_tcp_timeout_established = 36000
```

如果是 CentOS 5.X 系统，需执行 modprobe ip_conntrack 命令，然后在内核优化文件中加入如下代码，sysctl -p 使其内核文件生效，即可解决该报错。

```
net.ipv4.ip_conntrack_max = 655350
net.ipv4.netfilter.ip_conntrack_tcp_timeout_established = 10800
```

15.6 影响服务器性能因素

影响企业生产环境 Linux 服务器性能的因素有很多，一般分为两大类，即操作系统层级和应用程序级别。以下为各级别影响性能的具体项及性能评估的标准。

(1) 操作系统级别。

- 内存；
- CPU；
- 磁盘 I/O；
- 网络 I/O 带宽。

(2) 应用程序及软件。

- Nginx；
- MySQL；
- Tomcat；
- PHP；
- 应用程序代码。

(3) Linux 系统性能评估标准如表 15-1 所示。

表 15-1 Linux 性能评估标准

影响性能因素	评判标准		
	好	坏	糟 糕
CPU	user% + sys%< 70%	user% + sys%= 85%	user% + sys% ≥ 90%
内存	Swap In(si)=0 Swap Out(so)=0	Per CPU with 10 page/s	More Swap In & Swap Out
磁盘	iowait % < 20%	iowait % =35%	iowait % ≥ 50%

(4) Linux 系统性能分析工具。

常用系统性能分析命令为 vmstat、sar、iostat、netstat、free、ps、top、iftop 等。

常用系统性能组合分析命令如下：

- vmstat、sar、iostat：检测是否是 CPU 瓶颈。
- free、vmstat：检测是否是内存瓶颈。
- iostat：检测是否是磁盘 I/O 瓶颈。
- netstat、iftop：检测是否是网络带宽瓶颈。

15.7　Linux 服务器性能评估与优化

Linux 服务器性能评估与优化是一项长期的工作，需要随时关注网站服务器的运行状态，及时作出相应的调整，以下为 Linux 服务器性能评估及优化方案。

(1) Linux 系统整体性能评估。

uptime 命令主要用于查看当前服务器整体性能，例如 CPU、负载、内存等值的总览，以下为 uptime 命令应用案例及详解。

```
[root@web1 ~]# uptime
13:38:00 up 112 days, 14:01, 5 users, load average: 6.22, 1.02, 0.91
```

load average 负载有 3 个值，分别表示最近 1min、5min、15min 系统的负载，3 个值的大小一般不能大于系统逻辑 CPU 核数的 2 倍，例如 Linux 操作系统有 4 个逻辑 CPU，如果 load average 的 3 个值长期大于 8 时，说明 CPU 很繁忙，负载很高，可能会影响系统性能，但是偶尔大于 8 时，可以不用担心，一般不会影响系统性能。

如果 load average 的输出值小于 CPU 逻辑个数的 2 倍，则表示 CPU 还有空闲的时间片，例如案例中 CPU 负载为 6.22，表示 CPU 或者服务器是比较空闲的。基于此参数不能完全确认服务器的性能瓶颈，需要借助其他工具进一步判断。

(2) CPU 性能评估。

利用 vmstat 命令监控系统 CPU，该命令可以显示关于系统各种资源之间相关性能的简要信息，主要用它来查看 CPU 负载及队列情况。如图 15-6 所示，为 vmstat 命令在某个系统的输出结果。

```
[root@PT-171123 ~]# vmstat 2 10
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 12529508 372016 16116888    0    0     0     2    0    0  0  0 100  0  0
 0  0      0 12529520 372016 16116888    0    0     0    40  137  214  0  0 100  0  0
 0  0      0 12529520 372016 16116888    0    0     0     0  135  219  0  0 100  0  0
 0  0      0 12529520 372016 16116888    0    0     0     0  120  205  0  0 100  0  0
 0  0      0 12529556 372016 16116888    0    0     0     8  139  232  0  0 100  0  0
 0  0      0 12529556 372016 16116896    0    0     0     0  159  250  0  0 100  0  0
 0  0      0 12529556 372016 16116896    0    0     0    26  129  213  0  0 100  0  0
 0  0      0 12529556 372016 16116896    0    0     0     6  120  210  0  0 100  0  0
 0  0      0 12529556 372016 16116896    0    0     0     0  122  203  0  0 100  0  0
 0  0      0 12529556 372016 16116896    0    0     0     0  123  209  0  0 100  0  0
[root@PT-171123 ~]#
```

图 15-6　vmstat 工具查看系统 CPU 资源

vmstat 输出结果详解如下：

- r：该列表示运行和等待 CPU 时间片的进程数，这个值如果长期大于系统 CPU 的个数，说明 CPU 不足，需要增加 CPU。
- b：该列表示在等待资源的进程数，比如正在等待 I/O 或者内存交换等。
- us：该列显示了用户进程消耗的 CPU 时间百分比，us 的值比较高时，说明用户进程消耗的 CPU 时间多，但是如果长期大于 50%，就需要考虑优化程序或算法。
- sy：该列显示了内核进程消耗的 CPU 时间百分比，sy 的值较高时，说明内核消耗的

CPU 资源很多。

- us+sy：参考值为 80%，如果 us+sy 大于 80%说明可能存在 CPU 资源不足。

利用 sar 命令监控系统 CPU，sar 功能很强大，可以对系统的每个方面进行单独的统计，但是使用 sar 命令会增加系统开销，不过这些开销是可以评估的，对系统的统计结果不会有很大影响。如图 15-7 所示，为 sar 命令对某个系统的 CPU 统计输出。

```
[root@PT-171123 ~]# sar -u 2 10
Linux 2.6.32-279.el6.x86_64 (PT-171123.360buy.com)      01/22/2015      _x86_64_        (16 CPU)

03:04:51 PM     CPU     %user     %nice   %system   %iowait    %steal     %idle
03:04:53 PM     all      0.00      0.00      0.03      0.00      0.00     99.97
03:04:55 PM     all      0.00      0.00      0.03      0.00      0.00     99.97
03:04:57 PM     all      0.03      0.00      0.03      0.00      0.00     99.94
03:04:59 PM     all      0.00      0.00      0.03      0.00      0.00     99.97
03:05:01 PM     all      0.00      0.00      0.03      0.00      0.00     99.97
03:05:03 PM     all      0.00      0.00      0.03      0.00      0.00     99.97
03:05:05 PM     all      0.00      0.00      0.00      0.00      0.00    100.00
03:05:07 PM     all      0.03      0.00      0.03      0.00      0.00     99.94
03:05:09 PM     all      0.03      0.00      0.03      0.00      0.00     99.94
```

图 15-7　sar 工具查看系统 CPU 资源

sar 输出结果详解如下：

- %user：该列显示了用户进程消耗的 CPU 时间百分比。
- %nice：该列显示了运行正常进程所消耗的 CPU 时间百分比。
- %system：该列显示了系统进程消耗的 CPU 时间百分比。
- %iowait：该列显示了 I/O 等待所占用的 CPU 时间百分比。
- %idle：该列显示了 CPU 处在空闲状态的时间百分比。
- %steal：列显示了在内存相对紧张的环境下 pagein 强制对不同的页面进行的 steal 操作。

（3）内存性能评估。

利用 free 指令监控内存，free 是监控 Linux 内存使用状况最常用的指令，如图 15-8 所示为服务器内存使用情况。



```
You have mail in /var/spool/mail/root
[root@PT-171123 ~]#
[root@PT-171123 ~]# free -m
             total       used       free     shared    buffers     cached
Mem:         32097      19861      12235          0        363      15739
-/+ buffers/cache:       3759      28337
Swap:          499          0        499
[root@PT-171123 ~]#
```

图 15-8　free 查看系统内存情况

一般而言，服务器内存可以通过以下方法判断是否空余。

- 应用程序可用内存/系统物理内存大于 70%时，表示系统内存资源非常充足，不影响系统性能；
- 应用程序可用内存/系统物理内存小于 20%时，表示系统内存资源紧缺，需要增加系统内存；
- 20%<应用程序可用内存/系统物理内存小于 70%时，表示系统内存资源基本能满

足应用需求，暂时不影响系统性能。

(4) 磁盘 I/O 性能评估。

利用 iostat 评估磁盘性能，监控磁盘 I/O 读写及带宽，如图 15-9 所示。

```
[root@PT-171123 ~]# iostat -d 1 10
Linux 2.6.32-279.el6.x86_64 (PT-171123.360buy.com)     01/22/2015      _x86_64_        (16 CPU)

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               2.02         9.44        48.65   75171924  387376816

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               0.00         0.00         0.00          0          0

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               0.00         0.00         0.00          0          0

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               0.00         0.00         0.00          0          0

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               0.00         0.00         0.00          0          0

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               0.00         0.00         0.00          0          0

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               0.00         0.00         0.00          0          0

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               0.00         0.00         0.00          0          0
```

图 15-9 iostat 评估磁盘性能

iostat 输出结果详解如下：

- Blk_read/s：表示每秒读取的数据块数。
- Blk_wrtn/s：表示每秒写入的数据块数。
- Blk_read：表示读取的所有块数。
- Blk_wrtn：表示写入的所有块数。

可以通过 Blk_read/s 和 Blk_wrtn/s 的值对磁盘的读写性能有基本的了解，如果 Blk_wrtn/s 值很大，表示磁盘的写操作很频繁，可以考虑优化磁盘或者优化程序，如果 Blk_read/s 值很大，表示磁盘直接读取操作很多，可以将读取的数据放入内存中进行操作。

利用 sar 评估磁盘性能，通过 sar -d 组合，可以对系统的磁盘 I/O 做基本的统计，如图 15-10 所示。

```
[root@PT-171123 ~]# sar -d 1 10
Linux 2.6.32-279.el6.x86_64 (PT-171123.360buy.com)     01/22/2015      _x86_64_        (16 CPU)

03:26:41 PM       DEV       tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz     await     svctm     %util
03:26:42 PM    dev8-0      6.00      0.00    120.00     20.00      0.00      0.50      0.50      0.30

03:26:42 PM       DEV       tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz     await     svctm     %util
03:26:43 PM    dev8-0      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00

03:26:43 PM       DEV       tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz     await     svctm     %util
03:26:44 PM    dev8-0      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00

03:26:44 PM       DEV       tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz     await     svctm     %util
03:26:45 PM    dev8-0      2.00      0.00     32.00     16.00      0.00      0.00      0.00      0.00

03:26:45 PM       DEV       tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz     await     svctm     %util
03:26:46 PM    dev8-0      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00

03:26:46 PM       DEV       tps  rd_sec/s  wr_sec/s  avgrq-sz  avgqu-sz     await     svctm     %util
03:26:47 PM    dev8-0     13.86      0.00    110.89      8.00      0.00      0.00      0.00      0.00
```

图 15-10 sar 查看系统磁盘 I/O

sar 输出结果详解如下：

- await：表示平均每次设备 I/O 操作的等待时间（以 ms 为单位）。
- svctm：表示平均每次设备 I/O 操作的服务时间（以 ms 为单位）。
- %util 表示 1s 中有百分之几的时间用于 I/O 操作。

磁盘 I/O 性能，评判标准为正常情况下 svctm 应该是小于 await 值的，而 svctm 的大小和磁盘性能有关，CPU、内存的负荷也会对 svctm 值造成影响，过多的请求也会间接地导致 svctm 值的增加。

await 值的大小一般取决于 svctm 的值和 I/O 队列长度以及 I/O 请求模式，如果 svctm 的值与 await 很接近，表示几乎没有 I/O 等待，磁盘性能很好，如果 await 的值远高于 svctm 的值，则表示 I/O 队列等待太长，系统上运行的应用程序将变慢，此时可以通过更换更快的硬盘来解决问题。

%util 项的值也是衡量磁盘 I/O 的一个重要指标，如果%util 接近 100%，表示磁盘产生的 I/O 请求太多，I/O 系统已经满负荷的在工作，该磁盘可能存在瓶颈。长期下去，势必影响系统的性能，可以通过优化程序或者通过更换更高、更快的磁盘来解决此问题。

（5）网络性能评估。

- 通过 ping 命令检测网络的连通性；
- 通过 netstat -i 组合检测网络接口状况；
- 通过 netstat -r 组合检测系统的路由表信息；
- 通过 sar -n 组合显示系统的网络运行状态。

通过 iftop -i eth0 可以查看网卡流量，详细参数如下，详情如图 15-11 所示。

```
                        1.91Mb          3.81Mb          5.72Mb          7.63Mb     9.54Mb
host8.xirang.com           => 103.253.234.124.broad.cc.   29.9Kb   28.3Kb   28.3Kb
                           <=                              1.21Mb   1.05Mb   1.05Mb
host8.xirang.com           => 180.109.105.197                 0b    682Kb    682Kb
                           <=                                 0b   34.0Kb   34.0Kb
host8.xirang.com           => 117.43.86.155                1.85Mb    681Kb    681Kb
                           <=                              49.7Kb   24.6Kb   24.6Kb
host8.xirang.com           => 222.244.34.139                187Kb    257Kb    257Kb
                           <=                              10.8Kb   42.2Kb   42.2Kb
host8.xirang.com           => 122.242.117.169                 0b   94.0Kb   94.0Kb
                           <=                                 0b   21.4Kb   21.4Kb
host8.xirang.com           => 78.251.24.243                 152Kb    102Kb    102Kb
                           <=                              2.03Kb   1.61Kb   1.61Kb
host8.xirang.com           => 220.191.239.162              40.5Kb   71.6Kb   71.6Kb
                           <=                              4.96Kb   14.2Kb   14.2Kb
host8.xirang.com           => 123.87.97.224                  368b   27.2Kb   27.2Kb
                           <=                                416b   3.33Kb   3.33Kb
host8.xirang.com           => 202.108.251.188                 0b     850b     850b
                           <=                                 0b   16.1Kb   16.1Kb

TX:              cumm:  2.46MB   peak:   3.05Mb  rates:   2.28Mb   1.97Mb   1.97Mb
RX:                     1.52MB           1.51Mb           1.29Mb   1.22Mb   1.22Mb
TOTAL:                  3.98MB           4.20Mb           3.57Mb   3.19Mb   3.19Mb
```

图 15-11　iftop 查看系统网卡流量

- ＜＝：客户端流入的流量。
- ＝＞：服务器端流出的流量。
- TX：发送流量。
- RX：接收流量。
- TOTAL：总流量。
- cumm：运行 iftop 到目前时间的总流量。
- peak：流量峰值。
- rates：分别表示过去 2s、10s、40s 的平均流量。

第 16 章



大数据备份企业实战

随着互联网不断地发展，企业对运维人员的能力要求也越来越高，尤其是要求运维人员能处理各种故障、专研自动化运维技术、云计算机、虚拟化等，以满足公司业务的快速发展。

本章向读者介绍数据库备份方法、数量级 2TB 及以上级别数据库备份方案、xtrabackup 企业工具案例演示、数据库备份及恢复实战等内容。

16.1　企业级数据库备份实战

在日常的运维工作中，数据是公司非常重要的资源，尤其是数据库的相关信息，如果数据丢了，少则损失几千元，高则损失上千万。所以在运维工作中要及时注意网站数据的备份，要对数据库进行不定期的备份。

企业中如果数据量达到 TB 级别，维护和管理是非常复杂的，尤其是对数据库进行备份操作。

16.2　数据库备份方法及策略

企业中 MySQL 数据库备份最常用的方法如下：

- 直接 cp 备份；
- sqlhotcopy 备份；
- 主从同步复制；
- Mysqldump 备份；
- xtrabackup 备份。

mysqldump 和 xtrabackup 均可以备份 MySQL 数据，以下为 mysqldump 工具使用方法。

通常小于 100GB 的 MySQL 数据库可以使用默认 mysqldump 备份工具进行备份，如果是超过 100GB 的大数据，由于 mysqldump 备份方式是采用的逻辑备份，最大的缺陷是备份和恢复速度较慢。

基于 mysqldump 备份耗时会非常长，而且备份期间会锁表，锁表直接导致数据库只能访问 select，不能执行 insert、update 等操作，进而导致部分 Web 应用无法写入新数据。

如果是 myisam 引擎表，当然也可以执行参数--lock-tables＝false 禁用锁表，但是有可能造成数据信息不一致。

如果是支持事务的表，例如 innoDB 和 BDB，--single-transaction 参数是一个更好的选择，因为它不锁定表，具体应用如下：

```
mysqldump -uroot -p123456 --all-databases --opt --single-transaction > 2017all.sql
```

其中 --opt 快捷选项，等同于添加 --add-drop-tables --add-locking --create-option --disable-keys --extended-insert --lock-tables --quick --set-charset 选项。该选项能让 mysqldump 很快地导出数据，并且导出的数据能很快导回。该选项默认开启，但可以用 --skip-opt 禁用。

如果运行 mysqldump 没有指定 --quick 或 --opt 选项，则会将整个结果集中放在内存中。如果导出大数据库的话可能会导致内存溢出而异常退出。

16.3　xtrabackup 企业实战

MySQL 冷备、mysqldump、MySQL 热拷贝均不能实现对数据库进行增量备份。在实际环境中增量备份非常的实用，如果数据量小于 100GB，存储空间足够，可以每天进行完整备份，如果每天产生的数据量大，需要定制数据备份策略。例如每周日使用完整备份，周一到周六使用增量备份，或者每周六完整备份，周日到周五使用增量备份。

Percona-xtrabackup 是为实现增量备份而生的一款主流备份工具，xtrabackup 有两个主要工具，分别为 xtrabackup、innobackupex。

Percona-xtrabackup 是 Percona 公司开发的一个用于 MySQL 数据库物理热备的备份工具。支持 MySQL、Percona server 及 MariaDB，开源免费，是目前互联网数据库备份最主流的工具之一。

xtrabackup 只能备份 innoDB 和 xtraDB 两种数据引擎的表，而不能备份 MyISAM 数据表，innobackupex 1.5.1 则封装了 xtrabackup，是一个封装好的脚本，使用该脚本能同时备份处理 innoDB 和 MyISAM，但在处理 MyISAM 时需要加一个读锁。

xtrabackup 备份原理：innobackupex 在后台线程不断追踪 innoDB 的日志文件，然后复制 innoDB 的数据文件。数据文件复制完成之后，日志的复制线程也会结束。这样就得到了不在同一时间点的数据副本和开始备份以后的事务日志。完成上面的步骤之后，就可以使用 innoDB 崩溃恢复代码执行事务日志(redo log)，以达到数据的一致性。其备份优点如下：

- 备份速度快，物理备份更加可靠；
- 备份过程不会打断正在执行的事务，无须锁表；

- 能够基于压缩等功能节约磁盘空间和流量；
- 自动备份校验；
- 还原速度快；
- 可以流传将备份传输到另外一台机器上；
- 节约磁盘空间和网络带宽。

innobackupex 工具的备份过程原理，如图 16-1 所示。

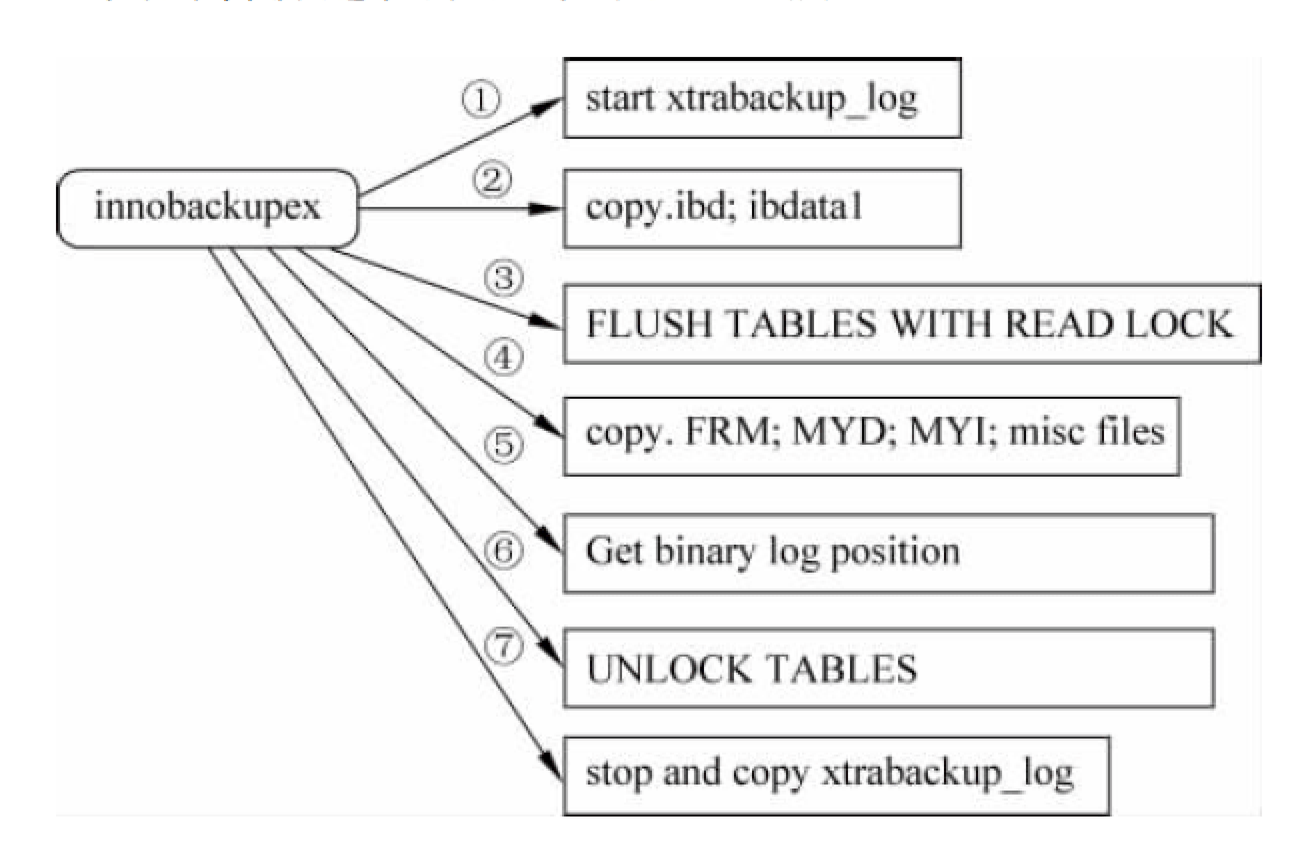


图 16-1 innobackupex 备份过程

innobackupex 备份过程中首先启动 xtrabackup_log 后台检测的进程，实时检测 MySQL redo 的变化，一旦发现 redo 有新的日志写入，立刻将日志写入到日志文件 xtrabackup_log 中，并复制 innoDB 的数据文件和系统表空间文件 idbdata1 到备份目录。

innode 引擎表备份完之后，执行 flush table with read lock 操作进行 MyLSAM 表备份。复制.frm .myd .myi 文件，并且在这一时刻获得 binary log 的位置，将表进行解锁 unlock tables，停止 xtrabackup_log 进程，完成整个数据库的备份。

16.4 Percona-xtrabackup 备份实战

基于 Percona-xtrabackup 备份，需要以下几个步骤。

(1) 官网下载 Percona-xtrabackup，Percona 官方 wiki 使用帮助如下。

```
http://www.percona.com/docs/wiki/percona-xtrabackup:start
wget
http://www.percona.com/redir/downloads/XtraBackup/XtraBackup-2.0.0/binary/Linux/x86_64/
percona-xtrabackup-2.0.0.tar.gz
```

(2) Percona-xtrabackup 软件安装方法，cp innobackupex、xtrabackup、xtrabackup_51、xtrabackup_55 工具到/usr/bin 目录下，代码如下：

```
tar zxvf percona - xtrabackup - 2.0.0.tar.gz
cp  percona - xtrabackup - 2.0.0/bin/innobackupex /usr/bin/innobackupex
cp  percona - xtrabackup - 2.0.0/bin/xtrabackup /usr/bin/xtrabackup
cp  percona - xtrabackup - 2.0.0/bin/xtrabackup_51 /usr/bin/xtrabackup_51
cp  percona - xtrabackup - 2.0.0/bin/xtrabackup_55 /usr/bin/xtrabackup_55
```

(3) MySQL 数据库全备份，代码如下，详情如图 16-2 所示。

```
innobackupex -- user = root -- password = 123456 /data/backup/mysql/
```

```
[root@localhost ~]#
[root@localhost ~]# ll /data/backup/mysql/
总用量 0
[root@localhost ~]#
[root@localhost ~]# innobackupex --user=root --password=123456 /data/backup/mysql/

InnoDB Backup Utility v1.5.1-xtrabackup; Copyright 2003, 2009 Innobase Oy
and Percona Inc 2009-2012.  All Rights Reserved.

This software is published under
the GNU GENERAL PUBLIC LICENSE Version 2, June 1991.

141220 00:10:18  innobackupex: Starting mysql with options:  --password=xxxxxxxx --user='r
141220 00:10:18  innobackupex: Connected to database with mysql child process (pid=7647)
141220 00:10:24  innobackupex: Connection to database server closed
IMPORTANT: Please check that the backup run completes successfully.
           At the end of a successful backup run innobackupex
           prints "completed OK!".
```

(a) innobackupex完整备份(1)

```
innobackupex: Backing up file '/var/lib/mysql//firstdb/first_t1.MYI'
innobackupex: Backing up file '/var/lib/mysql//firstdb/first_t1.frm'
innobackupex: Backing up file '/var/lib/mysql//firstdb/first_t1.MYD'
innobackupex: Backing up file '/var/lib/mysql//firstdb/db.opt'
innobackupex: Backing up files '/var/lib/mysql//nagios/*.{frm,MYD,MYI,MRG,TRG,TRN,ARM,
innobackupex: Backing up file '/var/lib/mysql//bbs/db.opt'
141220 00:10:47  innobackupex: Finished backing up .frm, .MRG, .MYD, .MYI, .TRG, .TRN,
les

innobackupex: Resuming ibbackup

xtrabackup: The latest check point (for incremental): '1606996'
xtrabackup: Stopping log copying thread.
.>> log scanned up to (1606996)

xtrabackup: Transaction log of lsn (1606996) to (1606996) was copied.
141220 00:10:49  innobackupex: All tables unlocked
141220 00:10:49  innobackupex: Connection to database server closed

innobackupex: Backup created in directory '/data/backup/mysql/2014-12-20_00-10-24'
innobackupex: MySQL binlog position: filename '', position
141220 00:10:49  innobackupex: completed OK!
[root@localhost ~]#
```

(b) innobackupex完整备份(2)

图 16-2 innobackupex 完整备份

(4) innobackupex 数据库恢复，恢复前先保证数据一致性，执行如下命令，详情如图 16-3 所示。

```
innobackupex -- defaults - file = /etc/my.cnf -- user = root -- password = 123456 -- apply -
log /data/backup/mysql/2014 - 12 - 20_00 - 10 - 24
```

```
[root@localhost ~]# ll /var/lib/mysql/
总用量 0
[root@localhost ~]#
[root@localhost ~]# cat /etc/my.cnf |grep datadir
datadir=/var/lib/mysql/
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# innobackupex --defaults-file=/etc/my.cnf --user=mysql --password=123456 --apply-log
l/2014-12-20_00-10-24/

InnoDB Backup Utility v1.5.1-xtrabackup; Copyright 2003, 2009 Innobase Oy
and Percona Inc 2009-2012.  All Rights Reserved.

This software is published under
```

图 16-3　innobackupex apply-log 恢复

通常数据库备份完成后，数据尚不能直接用于恢复操作，因为备份数据是一个过程，在备份过程中，有任务会写入数据，可能会包含尚未提交的事务或已经提交但尚未同步至数据文件中的事务。

因此此时数据文件仍处于不一致状态，基于--apply-log 可以通过回滚未提交的事务及同步已经提交的事务至数据文件使数据文件处于一致性状态，方可进行恢复数据。

apply-log 过程可以在任何机器上运行，没有强制在线上或者备份库上运行，可以把备份复制在闲置的服务器上去运行，以此来降低备份库的压力，但必须保证 backup 和 apply-log 所使用的 mysqlbackup 的版本要一致。

（5）删除原数据目录/var/lib/mysql 数据，使用参数--copy-back 恢复完整数据，授权 MySQL 用户给所有的数据库文件，代码如下，详情如图 16-4 所示。

```
rm -rf /var/lib/mysql/*
innobackupex --defaults-file=/etc/my.cnf --user=mysql --password=123456 --copy-
back /data/backup/mysql/2014-12-20_00-10-24/
chown -R mysql:mysql /var/lib/mysql/
```

```
drwxr-xr-x 2 root root   36864 12月 20 00:23 shareku
drwxr-xr-x 2 root root    4096 12月 20 00:23 test
drwxr-xr-x 2 root root    4096 12月 20 00:23 test01
drwxr-xr-x 2 root root    4096 12月 20 00:23 web02
drwxr-xr-x 2 root root    4096 12月 20 00:23 wordpress
drwxr-xr-x 2 root root    4096 12月 20 00:23 wugk1
-rw-r--r-- 1 root root      77 12月 20 00:23 xtrabackup_checkpoints
[root@localhost mysql]# chown -R mysql:mysql /var/lib/mysql/
[root@localhost mysql]#
[root@localhost mysql]#
[root@localhost mysql]# /usr/local/mysql/bin/mysqld_safe --user=mysql&
[1] 11335
[root@localhost mysql]# 141220 00:24:49 mysqld_safe Logging to '/var/log/my
141220 00:24:49 mysqld_safe Starting mysqld daemon with databases from /var

[root@localhost mysql]#
[root@localhost mysql]# ps -ef |grep mysql
root     11335  2948  0 00:24 pts/5    00:00:00 /bin/sh /usr/local/mysql/bi
mysql    11497 11335  0 00:24 pts/5    00:00:00 /usr/local/mysql/bin/mysqld
/mysql/ --plugin-dir=/usr/local/mysql/lib/plugin --user=mysql --log-error=/
mysqld.pid --socket=/var/lib/mysql/mysql.sock
root     11518  2948  0 00:24 pts/5    00:00:00 grep mysql
[root@localhost mysql]#
```

图 16-4　innobackupex 数据恢复

查看数据库恢复信息，数据完全恢复，如图 16-5 所示。

```
[root@localhost mysql]# mysql -uroot -p123456
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.5.20 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All ri

Oracle is a registered trademark of Oracle Corporation and/or
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| 122dsfds           |
| 122s               |
| asset              |
| audit              |
```

图 16-5　innobackupex 数据恢复

16.5　innobackupex 增量备份

增量备份仅能应用于 innoDB 或 xtraDB 表，对于 MyISAM 表而言，执行增量备份时实际上进行的是完全备份。

（1）增量备份之前必须执行完全备份，代码如下，详情如图 16-6 所示。

```
innobackupex --user=root --password=123456 --databases=wugk01 /data/backup/mysql/
```

```
innobackupex: Backup created in directory '/data/backup/mysql/2014-12-20_
innobackupex: MySQL binlog position: filename '', position
141220 13:02:02  innobackupex: completed OK!
[root@localhost mysql]# ls
2014-12-20_13-01-43
[root@localhost mysql]# clear
[root@localhost mysql]# ls
2014-12-20_13-01-43
[root@localhost mysql]# cd 2014-12-20_13-01-43/
[root@localhost 2014-12-20_13-01-43]# ls
backup-my.cnf  ibdata1  wugk01  xtrabackup_binary  xtrabackup_binlog_info
[root@localhost 2014-12-20_13-01-43]# pwd
/data/backup/mysql/2014-12-20_13-01-43
[root@localhost 2014-12-20_13-01-43]#
```

图 16-6　innobackupex 完整备份

（2）执行第一次增量备份，代码如下：

```
innobackupex --defaults-file=/etc/my.cnf --user=root --password=123456 --databases
=wugk01 --incremental /data/backup/mysql/ --incremental-basedir=/data/backup/mysql/
2014-12-20_13-01-43/
```

增量备份完后，会在/data/backup/mysql/目录下生成新的备份目录，如图 16-7 所示。

（3）数据库插入新数据，如图 16-8 所示。

```
xtrabackup: Transaction log of lsn (1613058) to (1613058) was copied.
141220 13:07:49  innobackupex: All tables unlocked
141220 13:07:49  innobackupex: Connection to database server closed

innobackupex: Backup created in directory '/data/backup/mysql/2014-12-20_13-07-31'
innobackupex: MySQL binlog position: filename '', position
141220 13:07:49  innobackupex: completed OK!
[root@localhost mysql]# ls
2014-12-20_13-01-43  2014-12-20_13-07-31
[root@localhost mysql]# du -sh *
19M     2014-12-20_13-01-43
256K    2014-12-20_13-07-31
[root@localhost mysql]#
```

图 16-7　innobackupex 增量备份

```
Database changed
mysql> insert into test_01 values('003','wuguangke3');
Query OK, 1 row affected (0.02 sec)

mysql> select * from test_01;
+------+------------+
| id   | name       |
+------+------------+
| 001  | wuguangke  |
| 002  | wuguangke2 |
| 003  | wuguangke3 |
+------+------------+
3 rows in set (0.00 sec)
```

图 16-8　数据库插入新数据

（4）执行第二次增量备份，代码如下，详情如图 16-9 所示。

```
innobackupex --defaults-file=/etc/my.cnf --user=root --password=123456 --databases
=wugk01 --incremental /data/backup/mysql/ --incremental-basedir=/data/backup/mysql/
2014-12-20_13-07-31/
```

```
[root@localhost mysql]#
[root@localhost mysql]# innobackupex --defaults-file=/etc/my.cnf --user=root --password=123456
mental /data/backup/mysql/ --incremental-basedir=/data/backup/mysql/2014-12-20_13-07-31/

InnoDB Backup Utility v1.5.1-xtrabackup; Copyright 2003, 2009 Innobase Oy
and Percona Inc 2009-2012.  All Rights Reserved.

This software is published under
the GNU GENERAL PUBLIC LICENSE Version 2, June 1991.

141220 13:11:14  innobackupex: Starting mysql with options:  --defaults-file='/etc/my.cnf' --p
ot' --unbuffered --
141220 13:11:14  innobackupex: Connected to database with mysql child process (pid=4094)
```

(a) 数据库增量备份(1)

```
141220 13:11:39  innobackupex: completed OK!
[root@localhost mysql]# innobackupex --defaults-file=/etc/my.cnf --user=ro
mental /data/backup/mysql/ --incremental-basedir=/data/backup/mysql/2014-1
[root@localhost mysql]# clear
[root@localhost mysql]# ls
2014-12-20_13-01-43  2014-12-20_13-07-31  2014-12-20_13-11-20
[root@localhost mysql]# du -sh *
19M     2014-12-20_13-01-43
256K    2014-12-20_13-07-31
224K    2014-12-20_13-11-20
[root@localhost mysql]#
```

(b) 数据库增量备份(2)

图 16-9　数据库增量备份

16.6　MySQL 增量备份恢复

删除原数据库中表及数据记录信息，如图 16-10 所示。

```
Database changed
mysql> show tables;
+------------------+
| Tables_in_wugk01 |
+------------------+
| test_01          |
+------------------+
1 row in set (0.00 sec)

mysql> drop table test_01;
Query OK, 0 rows affected (0.01 sec)

mysql> show tables;
Empty set (0.00 sec)
```

图 16-10　删除数据库表及数据记录信息

MySQL 增量备份数据恢复方法步骤如下。

(1) 基于 apply-log 确保数据一致性，代码如下：

```
innobackupex -- defaults - file = /etc/my. cnf -- user = root -- password = 123456 -- apply -
log -- redo - only /data/backup/mysql/2014 - 12 - 20_13 - 01 - 43/
```

(2) 执行第一次增量数据恢复，代码如下：

```
innobackupex -- defaults - file = /etc/my. cnf -- user = root -- password = 123456 -- apply -
log -- redo - only /data/backup/mysql/2014 - 12 - 20_13 - 01 - 43/ -- incremental - dir = /data/
backup/mysql/2014 - 12 - 20_13 - 07 - 31/
```

(3) 执行第二次增量数据恢复，代码如下：

```
innobackupex -- defaults - file = /etc/my. cnf -- user = root -- password = 123456 -- apply -
log -- redo - only /data/backup/mysql/2014 - 12 - 20_13 - 01 - 43/ -- incremental - dir = /data/
backup/mysql/2014 - 12 - 20_13 - 11 - 20/
```

(4) 执行完整数据恢复，代码如下：

```
innobackupex -- defaults - file = /etc/my. cnf -- user = root -- password = 123456 -- copy -
back /data/backup/mysql/2014 - 12 - 20_13 - 01 - 43/
```

(5) 测试数据库已完全恢复，如图 16-11 所示。

```
Database changed
mysql> show tables;
+------------------+
| Tables_in_wugk01 |
+------------------+
| test_01          |
+------------------+
1 row in set (0.00 sec)

mysql> select * from test_10;
ERROR 1146 (42S02): Table 'wugk01.test_10' doesn't exist
mysql> select * from test_01;
+------+------------+
| id   | name       |
+------+------------+
| 001  | wuguangke  |
| 002  | wuguangke2 |
| 003  | wuguangke3 |
+------+------------+
3 rows in set (0.00 sec)
```

图 16-11　数据库表信息完整恢复

第 17 章



shell 企业编程基础

说到 shell 编程，很多从事 Linux 运维工作的朋友都不陌生，都对 shell 有基本的了解，读者可能刚开始接触 shell 的时候，有各种想法，感觉编程非常困难，但 shell 编程是所有编程语言中最容易上手，最容易学习的编程脚本语言。

本章向读者介绍 shell 编程入门、shell 编程变量、if、while、for、case、select 基本语句案例演练及 shell 编程四剑客 find、grep、awk、sed 深度剖析等内容。

17.1 shell 编程入门简介

曾经有人说过，学习 Linux 不知道 shell 编程，那就是不懂 Linux，现在细细品味确实是这样。shell 是操作系统的最外层，shell 可以合并编程语言以控制进程和文件，以及启动和控制其他程序。

shell 通过提示您输入，向操作系统解释该输入，然后处理来自操作系统的任何结果输出，简单来说 shell 就是一个用户跟操作系统之间的一个命令解释器。

shell 是用户与 Linux 操作系统之间沟通的桥梁，用户可以输入命令执行，又可以利用 shell 脚本编程去运行，如图 17-1 所示。

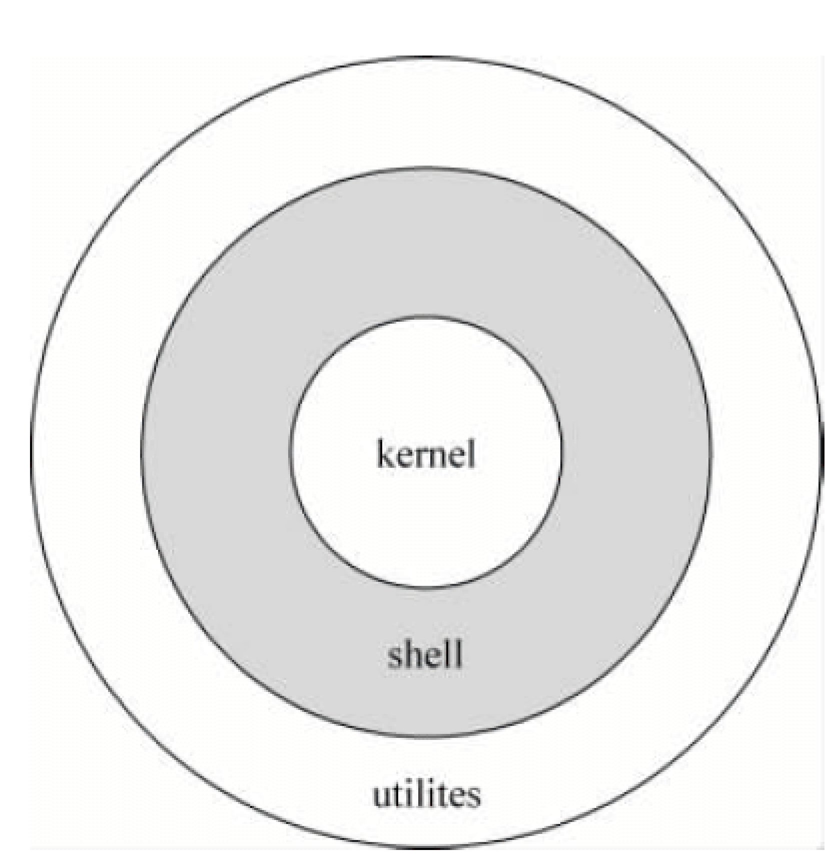


图 17-1　shell、utilites 及 kernel 位置关系

Linux shell 种类非常多，常见的 shell 如下：

- bourne shell(/usr/bin/sh 或/bin/sh)；
- bourne again shell(/bin/bash)；
- C shell(/usr/bin/csh)；
- K shell(/usr/bin/ksh)；
- shell for root(/sbin/sh)。

不同的 shell 语言的语法有所不同，一般不能交换使用，最常用的 shell 是 bash，也就是 bourne again shell。bash 由于易用和免费，在日常工作中被广泛使用，也是大多数 Linux 操作系统默认的 shell 环境。

shell、shell 编程、shell 脚本、shell 命令之间都有什么区别呢？简单来说：shell 是一个整体的概念，shell 编程与 shell 脚本统称为 shell 编程，shell 命令是 shell 编程底层具体的语句和实现方法。

17.2 shell 脚本及 Hello World

要熟练掌握 shell 编程语言，需要大量的练习，初学者可以用 shell 打印“Hello World”字符，寓意着开始新的启程！

shell 脚本编程需要注意以下几个事项：

- shell 脚本名称命名一般为英文的大写、小写；
- 不能使用特殊符号、空格来命名；
- shell 脚本后缀以.sh 结尾；
- 不建议 shell 命名为纯数字，一般以脚本功能命名；
- shell 脚本内容首行需以＃！/bin/bash 开头；
- shell 脚本中变量名称尽量使用大写字母，字母间不能使用“-”，可以使用“_”；
- shell 脚本变量名称不能以数字、特殊符号开头。

以下为第一个 shell 编程脚本，脚本名称为 first_shell.sh，代码如下：

```
#!/bin/bash
#This is my First shell
#By author jfedu.net 2017
echo "Hello World "
```

first_shell.sh 脚本内容详解如下：

- ＃！/bin/bash：固定格式，定义该脚本所使用的 shell 类型。
- ＃This is my First shell：＃号表示注释，没有任何的意义，shell 不会解析它。
- ＃By author jfedu.net 2017：表示脚本创建人，＃号表示注解。
- echo “Hello World ！” shell 脚本主命令，执行该脚本呈现的内容。

shell 脚本编写完毕，如果运行该脚本，运行用户需要有执行权限，可以使用 chmod o+x first_shell.sh 赋予可执行权限。然后./first_shell.sh 执行即可，还可以直接使用命令执行/bin/sh first_shell.sh 直接运行脚本，不需要执行权限，最终脚本执行显示效果一样。

初学者学习 shell 编程，可以将在 shell 终端运行的各种命令依次写入到脚本内容中，可以把 shell 脚本当成是 shell 命令的堆积。

17.3 shell 编程之变量详解

shell 属于非类型的解释型语言，在使用变量时不像 C++、JAVA 语言编程时需要事先声明变量，shell 给一个变量赋值，实际上就是定义了变量，在 Linux 支持的所有 shell 中，都

可以用赋值符号“＝”为变量赋值，shell 为弱类型语言，定义变量不需要声明类型，如果在使用时需要明确变量的类型，可以使用 declare 指定类型，declare 常见参数如下：

- ＋/－：“－”可用来指定变量的属性，“＋”为取消变量所设的属性。
- -f：仅显示函数。
- r：将变量设置为只读。
- x：指定的变量会成为环境变量，可供 shell 以外的程序来使用。
- i：指定类型为数值，字符串或运算式。

shell 编程中变量分为 3 种：系统变量、环境变量、用户变量，其中系统变量在对参数判断和命令返回值判断时使用，而环境变量则主要是在程序运行时需要设置，用户变量又称为局部变量，多使用在 shell 脚本内部或者临时局部。

shell 变量名在定义时，首个字符必须为字母(a～z，A～Z)，不能以数字开头，中间不能有空格，可以使用下画线“_”，不能使用“－”，也不能使用标点符号等。

例如定义变量 A＝jfedu. net，定义这样一个变量，A 为变量名，jfedu. net 是变量的值，变量名有格式规范，变量的值可以随意指定。变量定义完成，如需要引用变量，可以使用 $A。

var. sh 脚本内容如下：

```
#!/bin/bash
#By author jfedu.net 2017
A=123
echo "Printf variables is $A."
```

执行该 shell 脚本，结果将会显示 Printf variables is jfedu. net。shell 常见的系统变量、环境变量、用户变量详解如下。

(1) shell 编程常见系统变量如下：

- $0：当前脚本的名称。
- $n：当前脚本的第 n 个参数，n＝1，2，…，9。
- $*：当前脚本的所有参数(不包括程序本身)。
- $#：当前脚本的参数个数(不包括程序本身)。
- $?：命令或程序执行完后的状态，返回 0 表示执行成功。
- $$：程序本身的 PID 号。

(2) shell 编程常见环境变量如下：

- PATH：命令所示路径，以冒号为分割。
- HOME：打印用户家目录。
- SHELL：显示当前 shell 类型。
- USER：打印当前用户名。
- ID：打印当前用户 ID 信息。
- PWD：显示当前所在路径。

- TERM：打印当前终端类型。
- HOSTNAME：显示当前主机名。

（3）shell 编程用户变量如下：

- A=jfedu.net：自定义变量 A。
- N_SOFT=nginx-1.12.0.tar.gz：自定义变量 N_SOFT。
- BACK_DIR=/data/backup/：自定义变量 BACK_DIR。
- IP1=192.168.1.11：自定义变量 IP1。
- IP2=192.168.1.12：自定义变量 IP2。

创建 echo 打印菜单 shell 脚本，代码如下：

```
#!/bin/bash
#auto install httpd
#By author jfedu.net 2017
echo -e '\033[32m----------------------------\033[0m'
FILE=httpd-2.2.31.tar.bz2
URL=http://mirrors.cnnic.cn/apache/httpd/
PREFIX=/usr/local/apache2/
echo -e "\033[36mPlease Select Install Menu:\033[0m"
echo
echo "1)官方下载 Httpd 文件包."
echo "2)解压 apache 源码包."
echo "3)编译安装 Httpd 服务器."
echo "4)启动 HTTPD 服务器."
echo -e '\033[32m----------------------------\033[0m'
sleep 20
```

运行脚本，执行结果如图 17-2 所示。

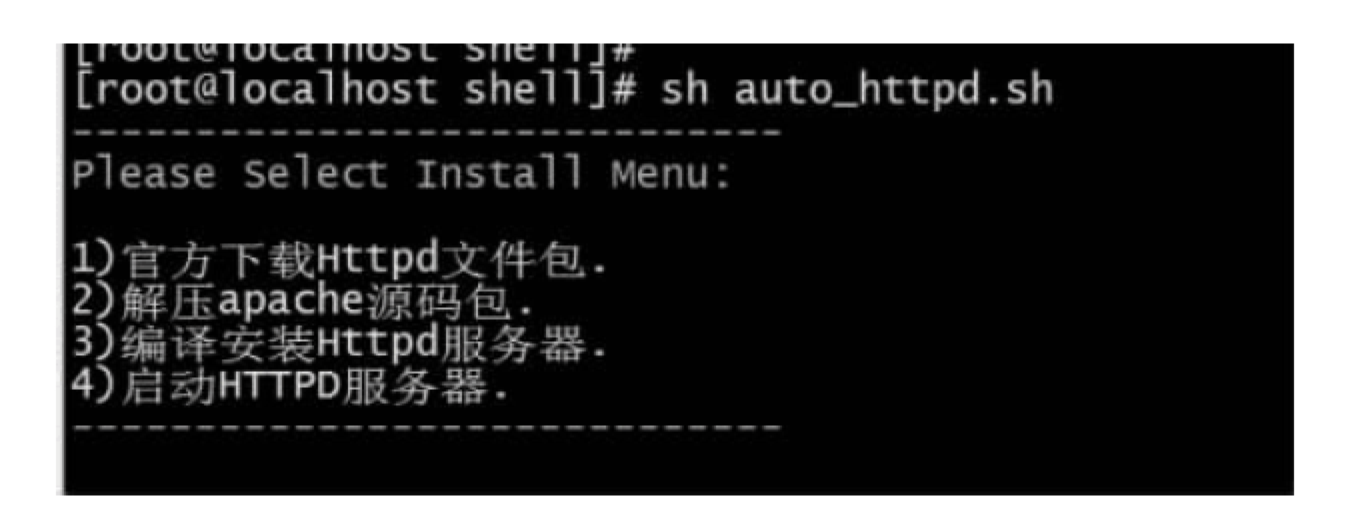


图 17-2　echo 打印菜单脚本

17.4　if 条件语句实战

Linux shell 编程中，if、for、while、case 等条件流程控制语句用得非常多，熟练掌握以上流程控制语句及语法的实战，对编写 shell 脚本有非常大的益处。

if 条件判断语句，通常以 if 开头，fi 结尾。也可加入 else 或者 elif 进行多条件的判断，if 表达式如下：

```
if (表达式)
    语句 1
else
    语句 2
fi
```

if 语句 shell 脚本编程案例如下。

(1) 比较两个整数大小，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
NUM = 100
if (( $NUM > 4 )) ; then
    echo "The Num $NUM more than 4."
else
    echo "The Num $NUM less than 4."
fi
```

(2) 判断系统目录是否存在，代码如下：

```
#!/bin/bash
#judge DIR or Files
#By author jfedu.net 2017
if [ ! -d /data/20140515 -a ! -d /tmp/2017/ ]; then
mkdir -p /data/20140515
```

if 常见的判断逻辑运算符详解如下：

- -f：判断文件是否存在，例如 if[-f filename]。
- -d：判断目录是否存在，例如 if[-d dir]。
- -eq：等于，应用于整型比较，即 equal。
- -ne：不等于，应用于整型比较，即 not equal。
- -lt：小于，应用于整型比较，即 letter。
- -gt：大于，应用于整型比较，即 greater。
- -le：小于或等于，应用于整型比较。
- -ge：大于或等于，应用于整型比较。
- -a：双方都成立(and)，用法为逻辑表达式-a 逻辑表达式。
- -o：单方成立(or)，用法为逻辑表达式-o 逻辑表达式。
- -z：空字符串。
- ||：单方成立。
- &&：双方都成立表达式。

(3) if 多个条件测试分数判断,代码如下:

```
#!/bin/bash
#By author jfedu.net 2017
scores= $1
if [[ $scores -eq 100 ]]; then
    echo "very good!";
elif [[ $scores -gt 85 ]]; then
    echo "good!";
elif [[ $scores -gt 60 ]]; then
    echo "pass!";
elif [[ $scores -lt 60 ]]; then
    echo "no pass!"
fi
```

17.5 if 判断括号区别

在使用 if 语句时,经常会使用()、(())、[]、[[]]、{}等括号,以下为几种括号简单区别对比。

- ():用于多个命令组、命令替换、初始化数组。
- (()):整数扩展、运算符、重定义变量值,算术运算比较。
- []:bash 内部命令,"["与 test 是等同的,正则字符范围、引用数组元素编号,不支持"+"、"-"、"*"、"/"数学运算符,逻辑测试使用-a、-o。
- [[]]:bash 程序语言的关键字,不是一个命令,[[]]结构比[]结构更加通用,不支持"+"、"-"、"*"、"/"数学运算符,逻辑测试使用 &&、||。
- {}:主要用于命令集合或者范围,例如 mkdir -p /data/201{7,8}/。

17.6 MySQL 数据库备份脚本

MySQL 数据库备份是运维工程师的工作之一,以下为自动备份 MySQL 数据库脚本,代码如下:

```
#!/bin/bash
#auto backup mysql
#By author jfedu.net 2017
#Define PATH 定义变量
BAK_DIR=/data/backup/mysql/'date +%Y-%m-%d'
MYSQLDB=webapp
MYSQLPW=backup
MYSQLUSR=backup
#must use root user run scripts 必须使用 root 用户运行,$UID 为系统变量
```

```
if
   [ $UID -ne 0 ]; then
   echo This script must use the root user ! ! !
   sleep 2
   exit 0
fi
#Define DIR and mkdir DIR 判断目录是否存在,不存在则新建
if
   [ ! -d $BAKDIR ]; then
   mkdir -p $BAKDIR
fi
#Use mysqldump backup Databases
/usr/bin/mysqldump -u $MYSQLUSR -p $MYSQLPW -d $MYSQLDB > $BAKDIR/webapp_db.sql
echo "The mysql backup successfully "
```

17.7 LAMP 一键自动化安装脚本

通过前面章节对 if 语句和变量的学习，接下来编写一键源码安装 LAMP 脚本，编写脚本要养成先分解脚本各个功能的习惯，这样有利于快速写出脚本，写出更高效的脚本。

一键源码安装 LAMP 脚本，可以拆分为如下功能：

(1) LAMP 打印菜单。

- 安装 Apache Web 服务器；
- 安装 MySQL DB 服务器；
- 安装 PHP 服务器；
- 整合 LAMP 架构；
- 启动 LAMP 服务。

(2) Apache 服务器安装部署。

Apache 官网下载 httpd-2.2.31.tar.gz 版本，解压，进入安装目录，执行 configure、make、make install 命令。

(3) MySQL 服务器的安装。

MySQL 官网下载 mysql-5.5.20.tar.gz 版本，解压，进入安装目录，执行 configure、make、make install 命令。

(4) PHP 服务器安装。

PHP 官网下载 php-5.3.8.tar.gz 版本，解压，进入安装目录，执行 configure、make、make install 命令。

(5) LAMP 整合及服务启动，代码如下：

```
vi /usr/local/apache2/htdocs/index.php
<?php
phpinfo();
```

```
?>
/usr/local/apache2/bin/apachectl restart
service mysqld restart
```

一键源码安装 LAMP 脚本，auto_install_lamp. sh 内容如下：

```
#!/bin/bash
#auto install LAMP
#By author jfedu.net 2017
#Httpd define path variable
H_FILES=httpd-2.2.31.tar.bz2
H_FILES_DIR=httpd-2.2.31
H_URL=http://mirrors.cnnic.cn/apache/httpd/
H_PREFIX=/usr/local/apache2/
#MySQL define path variable
M_FILES=mysql-5.5.20.tar.gz
M_FILES_DIR=mysql-5.5.20
M_URL=http://down1.chinaunix.net/distfiles/
M_PREFIX=/usr/local/mysql/
#PHP define path variable
P_FILES=php-5.3.28.tar.bz2
P_FILES_DIR=php-5.3.28
P_URL=http://mirrors.sohu.com/php/
P_PREFIX=/usr/local/php5/
echo -e '\033[32m-----------------------------\033[0m'
echo
if [ -z "$1" ]; then
    echo -e "\033[36mPlease Select Install Menu follow:\033[0m"
    echo -e "\033[32m
1)编译安装 Apache 服务器\033[1m"
    echo "2)编译安装 MySQL 服务器"
    echo "3)编译安装 PHP 服务器"
    echo "4)配置 index.php 并启动 LAMP 服务"
    echo -e "\033[31mUsage: { /bin/sh $0 1|2|3|4|help}\033[0m"
    exit
fi
if [[ "$1" -eq "help" ]]; then
        echo -e "\033[36mPlease Select Install Menu follow:\033[0m"
        echo -e "\033[32m1)编译安装 Apache 服务器\033[1m"
        echo "2)编译安装 MySQL 服务器"
        echo "3)编译安装 PHP 服务器"
        echo "4)配置 index.php 并启动 LAMP 服务"
        echo -e "\033[31mUsage: { /bin/sh $0 1|2|3|4|help}\033[0m"
        exit
fi
#Install httpd web server
if [[ "$1" -eq "1" ]]; then
```

```
    wget -c $H_URL/$H_FILES && tar -jxvf $H_FILES && cd $H_FILES_DIR &&./configure --
prefix= $H_PREFIX
    if [ $? -lt 0 ]; then
        make && make install
    fi
fi
#Install Mysql DB server
if [[ "$1" -eq "2" ]]; then
wget -c $M_URL/$M_FILES && tar -xzvf $M_FILES && cd $M_FILES_DIR &&yum install cmake -y ; cmake
. -DCMAKE_INSTALL_PREFIX= $M_PREFIX \
-DMYSQL_UNIX_ADDR=/tmp/mysql.sock \
-DMYSQL_DATADIR=/data/mysql \
-DSYSCONFDIR=/etc \
-DMYSQL_USER=mysql \
-DMYSQL_TCP_PORT=3306 \
-DWITH_XTRADB_STORAGE_ENGINE=1 \
-DWITH_INNOBASE_STORAGE_ENGINE=1 \
-DWITH_PARTITION_STORAGE_ENGINE=1 \
-DWITH_BLACKHOLE_STORAGE_ENGINE=1 \
-DWITH_MYISAM_STORAGE_ENGINE=1 \
-DWITH_READLINE=1 \
-DENABLED_LOCAL_INFILE=1 \
-DWITH_EXTRA_CHARSETS=1 \
-DDEFAULT_CHARSET=utf8 \
-DDEFAULT_COLLATION=utf8_general_ci \
-DEXTRA_CHARSETS=all \
-DWITH_BIG_TABLES=1 \
-DWITH_DEBUG=0
make && make install
/bin/cp support-files/my-small.cnf /etc/my.cnf
/bin/cp support-files/mysql.server /etc/init.d/mysqld
chmod +x /etc/init.d/mysqld
chkconfig --add mysqld
chkconfig mysqld on
if [ $? -eq 0 ]; then
    make && make install
    echo -e "\n\033[32m-----------------------------------------------\033[0m"
            echo -e "\033[32mThe $M_FILES_DIR Server Install Success !\033[0m"
        else
            echo -e "\033[32mThe $M_FILES_DIR Make or Make install ERROR,Please
            Check......"
            exit 0
        fi
fi
#Install PHP server
if [[ "$1" -eq "3" ]]; then
        wget -c $P_URL/$P_FILES && tar -jxvf $P_FILES && cd $P_FILES_DIR &&./configure --
```

```
prefix= $P_PREFIX --with-config-file-path= $P_PREFIX/etc --with-mysql= $M_PREFIX
 --with-apxs2= $H_PREFIX/bin/apxs
if [ $? -eq 0 ]; then
                make ZEND_EXTRA_LIBS='-liconv' && make install
                echo -e "\n\033[32m-----------------------------------\033[0m"
                echo -e "\033[32mThe $P_FILES_DIR Server Install Success !\033[0m"
        else
                echo -e "\033[32mThe $P_FILES_DIR Make or Make install ERROR,Please
                Check......"
                exit 0
        fi
fi
if [[ "$1" -eq "4" ]]; then
    sed -i '/DirectoryIndex/s/index.html/index.php index.html/g' $H_PREFIX/conf/httpd.conf
     $H_PREFIX/bin/apachectl restart
    echo "AddType     application/x-httpd-php .php" >> $H_PREFIX/conf/httpd.conf
    IP='ifconfig eth1|grep "Bcast"|awk '{print $2}'|cut -d: -f2'
    echo "You can access http://$IP/"
cat >$H_PREFIX/htdocs/index.php << EOF
<?php
phpinfo();
?>
EOF
fi
```

17.8 for 循环语句实战

for 循环语句主要用于对某个数据域进行循环读取、对文件进行遍历，通常用于循环某个文件或者列表。其语法格式以 for…do 开头，done 结尾。语法格式如下：

```
for var in (表达式)
do
    语句 1
done
```

for 循环语句 shell 脚本编程案例如下。

（1）循环打印 BAT 企业官网，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
for website in www.baidu.com www.taobao.com www.qq.com
do
    echo $website
done
```

(2) 循环打印 1～100 数字，seq 表示列出数据范围，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
for   i   in 'seq 1 100'
do
    echo "NUM is $i"
done
```

(3) for 循环求 1～100 的总和，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
#auto sum 1 100
j=0
for ((i=1; i<=100; i++))
do
     j='expr $i + $j'
done
echo $j
```

(4) 对系统日志文件进行分组打包，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
for   i   in 'find /var/log -name "*.log"'
do
    tar -czf 2017_log$i.tgz $i
done
```

(5) for 循环批量远程主机文件传输，代码如下：

```
#!/bin/bash
#auto scp files for client
#By author jfedu.net 2017
for i in 'seq 100 200'
do
    scp -r /tmp/jfedu.txt root@192.168.1.$i:/data/webapps/www
done
```

(6) for 循环批量远程主机执行命令，代码如下：

```
#!/bin/bash
#auto scp files for client
#By author jfedu.net 2017
for i in 'seq 100 200'
do
       ssh -l root 192.168.1.$i 'ls /tmp'
done
```

(7) for 循环打印 10s 等待提示,代码如下:

```
for ((j = 0; j <= 10; j++))
do
      echo  - ne "\033[32m - \033[0m"
      sleep 1
done
echo
```

17.9　while 循环语句实战

while 循环语句与 for 循环功能类似,主要用于对某个数据域进行循环读取、对文件进行遍历,通常用于循环某个文件或者列表,满足循环条件会一直循环,不满足则退出循环,其语法格式以 while…do 开头,done 结尾。语法格式如下:

```
while (表达式)
do
     语句 1
done
```

while 循环语句 shell 脚本编程案例如下。

(1) 循环打印 BAT 企业官网,read 指令用于读取行或者读取变量,代码如下:

```
#!/bin/bash
#By author jfedu.net 2017
while read line
do
     echo  $line
done < jfedu.txt
```

其中 jfedu.txt 内容如下:

```
www.baidu.com
www.taobao.com
www.qq.com
```

(2) while 无限每秒输出 Hello World,代码如下:

```
#!/bin/bash
#By author jfedu.net 2017
while sleep 1
do
     echo  - e "\033[32mHello World.\033[0m"
done
```

其中 jfedu.txt 内容如下：

```
www.baidu.com
www.taobao.com
www.qq.com
```

（3）循环打印 1～100 数字，expr 用于运算逻辑工具，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
i=0
while ((i<=100))
do
        echo $i
        i='expr $i + 1'
done
```

（4）while 循环求 1～100 的总和，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
#auto sum 1 100
j=0
i=1
while ((i<=100))
do
     j='expr $i + $j'
     ((i++))
done
echo $j
```

（5）while 循环逐行读取文件，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
while read line
do
    echo $line;
done</etc/hosts
```

（6）while 循环判断输入 IP 正确性，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
#Check IP Address
read -p "Please enter ip Address,example 192.168.0.11 ip": IPADDR
echo $IPADDR|grep -v "[Aa-Zz]"|grep --color -E "([0-9]{1,3}\.){3}[0-9]{1,3}"
while [ $? -ne 0 ]
do
```

```
        read -p "Please enter ip Address,example 192.168.0.11 ip": IPADDR
        echo $IPADDR|grep -v "[Aa-Zz]"|grep --color -E "([0-9]{1,3}\.){3}[0-9]{1,3}"
done
```

(7) 每 5s 循环判断/etc/passwd 是否被非法修改,代码如下:

```
#!/bin/bash
#Check File to change
#By author jfedu.net 2017
FILES="/etc/passwd"
while true
do
        echo "The Time is 'date +%F-%T'"
        OLD='md5sum $FILES|cut -d" " -f 1'
        sleep 5
        NEW='md5sum $FILES|cut -d" " -f 1'
        if [[ $OLD != $NEW ]]; then
            echo "The $FILES has been modified."
        fi
done
```

(8) 每 10s 循环判断 jfedu 用户是否登录系统,代码如下:

```
#!/bin/bash
#Check File to change.
#By author jfedu.net 2017
USERS="jfedu"
while true
do
        echo "The Time is 'date +%F-%T'"
        sleep 10
        NUM='who|grep "$USERS"|wc -l'
        if [[ $NUM -ge 1 ]]; then
                echo "The $USERS is login in system."
        fi
done
```

17.10 case 选择语句实战

case 选择语句主要用于对多个选择条件进行匹配输出,与 if…elif 语句结构类似,通常用于脚本传递输入参数,打印出输出结果及内容,其语法格式以 case…in 开头,esac 结尾。语法格式如下:

```
#!/bin/bash
#By author jfedu.net 2017
case $1 in
```

```
    Pattern1)
    语句 1
    ;;
    Pattern2)
    语句 2
    ;;
    Pattern3)
    语句 3
    ;;
esac
```

case 条件语句 shell 脚本编程案例如下。

（1）打印 monitor 及 archive 选择菜单，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
case $1 in
        monitor)
        monitor_log
        ;;
        archive)
        archive_log
        ;;
        help )
        echo -e "\033[32mUsage:{ $0 monitor | archive |help }\033[0m"
        ;;
        * )
        echo -e "\033[32mUsage:{ $0 monitor | archive |help }\033[0m "
esac
```

（2）自动修改 IP 脚本菜单，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
case $i in
                modify_ip)
                change_ip
                ;;
                modify_hosts)
                change_hosts
                ;;
                exit)
                exit
                ;;
                * )
                echo -e "1) modify_ip\n2) modify_ip\n3)exit"
esac
```

17.11　select 选择语句实战

select 语句一般用于选择，常用于选择菜单的创建，可以配合 PS3 来做打印菜单的输出信息，其语法格式以 select…in do 开头，done 结尾，语法格式如下：

```
select i in (表达式)
do
    语句
done
```

select 选择语句 shell 脚本编程案例如下。

（1）打印开源操作系统选择，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
PS3 = "What you like most of the open source system?"
select i in CentOS RedHat Ubuntu
do
    echo "Your Select System: " $i
done
```

（2）打印 LAMP 选择菜单，代码如下：

```
#!/bin/bash
#By author jfedu.net 2017
PS3 = "Please enter you select install menu:"
select i in http php mysql quit
do
case  $i in
        http)
        echo Test Httpd
        ;;
        php)
        echo Test PHP
        ;;
        mysql)
        echo Test MySQL
        ;;
        quit)
        echo The System exit
        exit
esac
done
```

17.12 shell 编程函数实战

shell 允许将一组命令集或语句形成一个可用块，这些块称为 shell 函数，shell 函数的好处在于只需定义一次，后期随时使用，无须在 shell 脚本中添加重复的语句块，其语法格式为以"function name(){"开头，以"}"结尾。

shell 编程函数默认不能将参数传入()内部，shell 函数参数传递在调用函数名称时传递，例如 name argv1 argv2，具体代码如下：

```
function name (){
        command1
        command2
        ...
}
name argv1 argv2
```

（1）创建 Apache 软件安装函数，给函数 Apache_install 传递参数 1，代码如下：

```
#!/bin/bash
#auto install LAMP
#By author jfedu.net 2017
#Httpd define path variable
H_FILES = httpd - 2.2.31.tar.bz2
H_FILES_DIR = httpd - 2.2.31
H_URL = http://mirrors.cnnic.cn/apache/httpd/
H_PREFIX = /usr/local/apache2/
function Apache_install()
{
#Install httpd web server
if [[ "$1" -eq "1" ]]; then
    wget -c $H_URL/$H_FILES && tar -jxvf $H_FILES && cd $H_FILES_DIR &&./configure --prefix
=$H_PREFIX
    if [ $? -eq 0 ]; then
        make && make install
        echo -e "\n\033[32m-----------------------------------------\033[0m"
        echo -e "\033[32mThe $H_FILES_DIR Server Install Success !\033[0m"
    else
        echo -e "\033[32mThe $H_FILES_DIR Make or Make install ERROR,Please Check......"
        exit 0
    fi
fi
}
Apache_install 1
```

（2）创建 judge_ip 判断 IP 函数，代码如下：

```
#!/bin/bash
```

```
#By author jfedu.net 2017
judge_ip(){
        read -p "Please enter ip Address,example 192.168.0.11 ip": IPADDR
        echo $IPADDR|grep -v "[Aa-Zz]"|grep --color -E "([0-9]{1,3}\.){3}[0-9]{1,3}"
}
judge_ip
```

17.13 shell 编程四剑客之 find

通过如上基础语法的学习，读者对 shell 编程有了更进一步的理解，shell 编程不再是简单命令的堆积，而是演变成了各种特殊的语句、各种语法、编程工具、各种命令的集合。

在 shell 编程工具中，四剑客工具的使用更加广泛，shell 编程四剑客包括 find、sed、grep、awk，熟练掌握四剑客会对 shell 编程能力有极大地提升。

四剑客之 find 工具实战，find 工具主要用于操作系统文件、目录的查找，其语法参数格式如下：

```
find   path   -option   [   -print ]   [ -exec   -ok   command ]   { } \;
```

其中 option 常用参数详解如下：

- -name filename：查找名为 filename 的文件。
- -type b/d/c/p/l/f：查找块设备、目录、字符设备、管道、符号链接、普通文件。
- -size n[c]：查找长度为 n 块[或 n 字节]的文件。
- -perm：按执行权限来查找。
- -user username：按文件属主来查找。
- -group groupname：按组来查找。
- -mtime −n ＋n：按文件更改时间来查找文件，−n 指 n 天以内，＋n 指 n 天以前。
- -atime −n ＋n：按文件访问时间来查找文件。
- -ctime −n ＋n：按文件创建时间来查找文件。
- -mmin −n ＋n：按文件更改时间来查找文件，−n 指 n 分钟以内，＋n 指 n 分钟以前。
- -amin −n ＋n：按文件访问时间来查找文件。
- -cmin −n ＋n：按文件创建时间来查找文件。
- -nogroup：查找无有效属组的文件。
- -nouser：查找无有效属主的文件。
- -newer f1 ！ f2：查找文件，−n 指 n 天以内，＋n 指 n 天以前。
- -depth：在进入子目录前先行查找完本目录。
- -fstype：查找更改时间比 f1 新但比 f2 旧的文件。

- -mount：查找文件时不跨越文件系统 mount 点。
- -follow：如果遇到符号链接文件，就跟踪链接所指的文件。
- -cpio：查找位于某一类型文件系统中的文件。
- -prune：忽略某个目录。
- -maxdepth：查找目录级别深度。

（1）find 工具-name 参数案例，详解如下：

- find /data/ -name "*.txt"：查找/data/目录以.txt 结尾的文件。
- find /data/ -name "[A-Z]*"：查找/data/目录以大写字母开头的文件。
- find /data/ -name "test*"：#查找/data/目录以 test 开头的文件。

（2）find 工具-type 参数案例，详解如下：

- find /data/ -type d：查找/data/目录下的文件夹。
- find /data/ ! -type d：查找/data/目录下的非文件夹。
- find /data/ -type l：查找/data/目录下的链接文件。
- find /data/ -type d|xargs chmod 755 -R：查找目录类型并将权限设置为 755。
- find /data/ -type f|xargs chmod 644 -R：查找文件类型并将权限设置为 644。

（3）find 工具-size 参数案例，详解如下：

- find /data/ -size +1M：查找文件大小大于 1MB 的文件。
- find /data/ -size 10M：查找文件大小为 10MB 的文件。
- find /data/ -size -1M：查找文件大小小于 1MB 的文件。

（4）find 工具-perm 参数案例，详解如下：

- find /data/ -perm 755：查找/data/目录权限为 755 的文件或者目录。
- find /data/ -perm -007：与-perm 777 相同，表示所有权限。
- find /data/ -perm +644：查找文件权限符号为 644 以上的文件。

（5）find 工具-mtime 参数案例，详解如下：

- atime，access time：文件被读取或者执行的时间。
- ctime，change time：文件状态改变时间。
- mtime，modify time：文件内容被修改的时间。
- find /data/ -mtime +30-name "*.log"：查找 30 天以前的 log 文件。
- find /data/ -mtime -30-name "*.txt"：查找 30 天以内的 log 文件。
- find /data/ -mtime 30-name "*.txt"：查找第 30 天的 log 文件。
- find /data/ -mmin +30-name "*.log"：查找 30min 以前修改的 log 文件。
- find /data/ -amin -30-name "*.txt"：查找 30min 以内被访问的 log 文件。
- find /data/ -cmin 30-name "*.txt"：查找第 30min 改变的 log 文件。

（6）find 工具参数综合案例，代码如下：

```
#查找/data 目录以.log 结尾,文件大于 10KB 的文件,同时 cp 到/tmp 目录
find /data/ -name "*.log" -type f -size +10k -exec cp {} /tmp/ \;
#查找/data 目录以.txt 结尾,文件大于 10KB 的文件,权限为 644 并删除该文件
find /data/ -name "*.log" -type f -size +10k -m perm 644 -exec rm -rf {} \;
#查找/data 目录以.log 结尾,30 天以前的文件,大小大于 10MB 并移动到/tmp 目录
find /data/ -name "*.log" -type f -mtime +30 -size +10M -exec mv {} /tmp/ \;
```

17.14　shell 编程四剑客之 sed

sed 是一个非交互式文本编辑器，它可对文本文件和标准输入进行编辑，标准输入可以来自键盘输入、文本重定向、字符串、变量，甚至来自于管道的文本，与 vim 编辑器类似，它一次处理一行内容，sed 可以编辑一个或多个文件，简化对文件的反复操作、编写转换程序等。

在处理文本时把当前处理的行存储在临时缓冲区中，称为“模式空间”(pattern space)，紧接着用 sed 命令处理缓冲区中的内容，处理完成后把缓冲区的内容输出至屏幕或者写入文件。逐行处理直到文件末尾，然而如果打印在屏幕上，实质文件内容并没有改变，除非用户使用重定向存储输出或者写入文件。其语法参数格式如下：

```
sed    [-Options]    ['Commands']    filename;
```

sed 工具默认处理文本，文本内容输出屏幕已经修改，但是文件内容其实没有修改，需要加-i 参数，即对文件彻底修改。具体参数详解如下：

- x：指定行号。
- x,y：指定从 x 到 y 的行号范围。
- /pattern/：查询包含模式的行。
- /pattern/pattern/：查询包含两个模式的行。
- /pattern/,x：从与 pattern 的匹配行到 x 号行之间的行。
- x,/pattern/：从 x 号行到与 pattern 的匹配行之间的行。
- x,y!：查询不包括 x 和 y 行号的行。
- r：从另一个文件中读文件。
- w：将文本写入到一个文件。
- y：变换字符。
- q：第一个模式匹配完成后退出。
- l：显示与八进制 ASCII 码等价的控制字符。
- {}：在定位行执行的命令组。
- p：打印匹配行。
- =：打印文件行号。
- a\：在定位行号之后追加文本信息。

- i\：在定位行号之前插入文本信息。
- d：删除定位行。
- c\：用新文本替换定位文本。
- s：使用替换模式替换相应模式。
- n：读取下一个输入行，用下一个命令处理新的行。
- N：将当前读入行的下一行读取到当前的模式空间。
- h：将模式缓冲区的文本复制到保持缓冲区。
- H：将模式缓冲区的文本追加到保持缓冲区。
- x：互换模式缓冲区和保持缓冲区的内容。
- g：将保持缓冲区的内容复制到模式缓冲区。
- G：将保持缓冲区的内容追加到模式缓冲区。

常用 sed 工具企业演练案例如下。

（1）替换 jfedu.txt 文本中 old 为 new，代码如下：

```
sed    's/old/new/g'        jfedu.txt
```

（2）打印 jfedu.txt 文本第一行至第三行，代码如下：

```
sed    -n '1,3p'            jfedu.txt
```

（3）打印 jfedu.txt 文本中第一行与最后一行，代码如下：

```
sed    -n '1p; $p'          jfedu.txt
```

（4）删除 jfedu.txt 第一行至第三行、删除匹配行至最后一行，代码如下：

```
sed    '1,3d'               jfedu.txt
sed    '/jfedu/, $d'        jfedu.txt
```

（5）删除 jfedu.txt 最后 6 行及删除最后一行，代码如下：

```
for    i in 'seq 1 6'; do sed -i   '$d' jfedu.txt ; done
sed    '$d'                        jfedu.txt
```

（6）删除 jfedu.txt 最后 6 行，代码如下：

```
sed    '$d'                 jfedu.txt
```

（7）在 jfedu.txt 查找 jfedu 所在行，并在其下一行添加 word 字符，a 表示在其下一行添加字符串，代码如下：

```
sed    '/jfedu/aword'       jfedu.txt
```

（8）在 jfedu.txt 查找 jfedu 所在行，并在其上一行添加 word 字符，i 表示在其上一行添加字符串，代码如下：

```
sed    '/jfedu/iword'      jfedu.txt
```

(9) 在 jfedu.txt 查找以.test 结尾的行，在其行尾添加字符串 word，$ 表示结尾标识，& 在 sed 中表示添加，代码如下：

```
sed    's/test$/&word/g'   jfedu.txt
```

(10) 在 jfedu.txt 查找 www 的行，在其行首添加字符串 word，^ 表示起始标识，& 在 sed 中表示添加，代码如下：

```
sed    '/www/s/^/&word/'   jfedu.txt
```

(11) 多个 sed 命令组合，使用-e 参数，代码如下：

```
sed -e '/www.jd.com/s/^/&1./' -e 's/www.jd.com$/&./g' jfedu.txt
```

(12) 多个 sed 命令组合，使用分号“;”分割，代码如下：

```
sed -e '/www.jd.com/s/^/&1./; s/www.jd.com$/&./g' jfedu.txt
```

(13) sed 读取系统变量，进行变量替换，代码如下：

```
WEBSITE=WWW.JFEDU.NET
sed "s/www.jd.com/$WEBSITE/g" jfedu.txt
```

(14) 修改 SELinux 策略 enforcing 为 disabled，查找/SELINUX/行，然后将其行 enforcing 值改成 disabled，!s 表示不包括 SELINUX 行，代码如下：

```
sed -i   '/SELINUX/s/enforcing/disabled/g' /etc/selinux/config
sed -i   '/SELINUX/!s/enforcing/disabled/g' /etc/selinux/config
```

通常而言，sed 将待处理的行读入模式空间，脚本中的命令逐行进行处理，直到脚本执行完毕，然后该行被输出，模式空间清空，然后重复刚才的动作，文件中的新的一行被读入，直到文件处理完备。

如果用户希望在某个条件下脚本中的某个命令被执行，或者希望模式空间得到保留以便下一次的处理，都有可能使得 sed 在处理文件的时候不按照正常的流程来进行。这时可以使用 sed 高级语法来满足用户需求。总的来说，sed 高级命令可以分为以下 3 种功能。

- N、D、P：处理多行模式空间的问题。
- H、h、G、g、x：将模式空间的内容放入存储空间以便接下来的编辑。
- :、b、t：在脚本中实现分支与条件结构。

(1) 在 jfedu.txt 每行后加入空行，也即每行占用两行空间，每一行后边插入一行空行、两行空行及前三行每行后插入空行，代码如下：

```
sed    '/^$/d; G'        jfedu.txt
sed    '/^$/d; G; G'     jfedu.txt
sed    '/^$/d; 1,3G; '   jfedu.txt
```

（2）将 jfedu.txt 偶数行删除及隔两行删除一行，代码如下：

```
sed     'n; d'            jfedu.txt
sed     'n; n; d'         jfedu.txt
```

（3）在 jfedu.txt 匹配行前一行、后一行插入空行以及同时在匹配前后插入空行，代码如下：

```
sed     '/jfedu/{x; p; x; }'  jfedu.txt
sed     '/jfedu/G'            jfedu.txt
sed     '/jfedu/{x; p; x; G; }jfedu.txt
```

（4）在 jfedu.txt 每行后加入空行，也即每行占用两行空间，每一行后边插入空行，代码如下：

```
sed '/^ $/d; G' jfedu.txt
```

（5）在 jfedu.txt 每行后加入两行空行，也即每行占用三行空间，每一行后边插入空行，代码如下：

```
sed '/^ $/d; G; G' jfedu.txt
```

（6）在 jfedu.txt 每行前加入顺序数字序号、加上制表符“\t”及“.”符号，代码如下：

```
sed = jfedu.txt| sed 'N; s/\n/ /'
sed = jfedu.txt| sed 'N; s/\n/\t/'
sed = jfedu.txt| sed 'N; s/\n/\./'
```

（7）删除 jfedu.txt 行前和行尾的任意空格，代码如下：

```
sed 's/^[ \t]*//; s/[ \t]* $//' jfedu.txt
```

（8）打印 jfedu.txt 关键词 old 与 new 之间的内容，代码如下：

```
sed -n '/old/,/new/'p     jfedu.txt
```

（9）打印及删除 jfedu.txt 最后两行，代码如下：

```
sed     '$!N; $!D'        jfedu.txt
sed     'N; $!P; $!D; $d' jfedu.txt
```

（10）合并上、下两行，也即两行合并，代码如下：

```
sed     '$!N; s/\n/ /'    jfedu.txt
sed     'N; s/\n/ /'      jfedu.txt
```

17.15 shell 编程四剑客之 awk

awk 是一个优良的文本处理工具，Linux 及 UNIX 环境中现有的功能最强大的数据处理引擎之一，以 Aho、Weinberger、Kernighan 三位发明者名字首字母命名为 awk，awk 是一

个行级文本高效处理工具，awk 经过改进生成的新的版本有 nawk、gawk，一般 Linux 默认为 gawk，gawk 是 awk 的 GNU 开源免费版本。

awk 基本原理是逐行处理文件中的数据，查找与命令行中所给定内容相匹配的模式，如果发现匹配内容，则进行下一个编程步骤，如果找不到匹配内容，则继续处理下一行。其语法参数格式如下：

```
awk    'pattern   +   {action}'    file
```

awk 常用参数、变量、函数详解如下。

(1) awk 基本语法参数详解如下：

- 单引号' '是为了和 shell 命令区分开。
- 大括号{ }表示一个命令分组。
- pattern 是一个过滤器，表示匹配 pattern 条件的行才进行 action 处理。
- action 是处理动作，常见动作为 print。
- 使用#作为注释，pattern 和 action 可以只有其一，但不能两者都没有。

(2) awk 内置变量详解如下：

- FS：分隔符，默认是空格。
- OFS：输出分隔符。
- NR：当前行数，从 1 开始。
- NF：当前记录字段个数。
- $0：当前记录。
- $1～$n：当前记录第 n 个字段(列)。

(3) awk 内置函数详解如下：

- gsub(r,s)：在 $0 中用 s 代替 r。
- index(s,t)：返回 s 中 t 的第一个位置。
- length(s)：s 的长度。
- match(s,r)：s 是否匹配 r。
- split(s,a,fs)：在 fs 上将 s 分成序列 a。
- substr(s,p)：返回 s 从 p 开始的子串。

(4) awk 常用操作符、运算符及判断符，详解如下：

- ++ --：增加与减少(前置或后置)。
- ^ **：指数(右结合性)。
- ! + -：非、一元(unary)加号、一元减号。
- + - * / %：加、减、乘、除、余数。
- < <= == != > >=：数字比较。
- &&：逻辑 and。
- ||：逻辑 or。

- = += -= *= /= %= ^= **=：赋值。

(5) awk 与流程控制语句如下：

- if(condition) { } else { }；
- while { }；
- do{ }while(condition)；
- for(init；condition；step){ }；
- break/continue。

常用 awk 工具企业演练案例。

(1) awk 打印硬盘设备名称，默认以空格为分割，代码如下：

```
df    -h|awk '{print $1}'
```

(2) awk 以空格、冒号、\t、分号为分割，代码如下：

```
awk -F '[ :\t; ]' '{print $1}'          jfedu.txt
```

(3) awk 以冒号分割，打印第一列，同时将内容追加到/tmp/awk.log 下，代码如下：

```
awk -F: '{print $1 >>"/tmp/awk.log"}'    jfedu.txt
```

(4) 打印 jfedu.txt 文件中的第 3 行至第 5 行，NR 表示打印行，$0 表示文本所有域，代码如下：

```
awk 'NR==3,NR==5 {print}'               jfedu.txt
awk 'NR==3,NR==5 {print $0}'            jfedu.txt
```

(5) 打印 jfedu.txt 文件中的第 3 行至第 5 行的第一列与最后一列，代码如下：

```
awk 'NR==3,NR==5 {print $1, $NF}'       jfedu.txt
```

(6) 打印 jfedu.txt 文件中长度大于 80 的行号，代码如下：

```
awk   'length( $0)>80 {print NR}'       jfedu.txt
```

(7) awk 引用 shell 变量，使用-v 或者双引号+单引号即可，代码如下：

```
awk -v STR=hello '{print STR, $NF}'     jfedu.txt
STR="hello"; echo| awk '{print "'${STR}'"; }'
```

(8) awk 以冒号切割，打印第一列同时只显示前 5 行，代码如下：

```
cat /etc/passwd|head -5|awk -F:   '{print $1}'
awk -F: 'NR>=1&&NR<=5 {print $1}' /etc/passwd
```

(9) awk 指定文件 jfedu.txt 第一列的总和，代码如下：

```
cat jfedu.txt |awk '{sum+= $1}END{print sum}'
```

(10) awk NR 行号除以 2 余数为 0 则跳过该行，继续执行下一行，打印在屏幕，代码

如下：

```
awk -F: 'NR%2==0 {next} {print NR, $1}' /etc/passwd
```

（11）awk 添加自定义字符，代码如下：

```
ifconfig eth0|grep "Bcast"|awk '{print "ip_" $2}'
```

（12）awk 格式化输出 passwd 内容，printf 打印字符串，%格式化输出分隔符，s 表示字符串类型，-12 表示 12 个字符，-6 表示 6 个字符，代码如下：

```
awk -F: '{printf "%-12s %-6s %-8s\n", $1, $2, $NF}' /etc/passwd
```

（13）awk OFS 输出格式化\t，代码如下：

```
netstat -an|awk '$6 ~ /LISTEN/&&NR>=1&&NR<=10 {print NR, $4, $5, $6}' OFS="\t"
```

（14）awk 与 if 组合实战，判断数字比较，代码如下：

```
echo 3 2 1 | awk '{ if(($1>$2) || ($1>$3)) { print $2} else {print $1} }'
```

（15）awk 与数组组合实战，统计 passwd 文件用户数，代码如下：

```
awk -F ':' 'BEGIN {count=0; } {name[count] = $1; count++; }; END{for (i = 0; i < NR; i++)
print i, name[i]}' /etc/passwd
```

（16）awk 分析 Nginx 访问日志的状态码 404、502 等错误信息页面，统计次数大于 20 的 IP 地址，代码如下：

```
awk '{if ($9~/502|499|500|503|404/) print $1, $9}' access.log|sort|uniq -c|sort -nr |awk '
{if($1>20) print $2}'
```

（17）用/etc/shadow 文件中的密文部分替换/etc/passwd 中的“x”位置，生成新的/tmp/passwd 文件，代码如下：

```
awk 'BEGIN{OFS=FS=":"} NR==FNR{a[$1]=$2}NR>FNR{$2=a[$1]; print >>"/tmp/passwd"}' /
etc/shadow /etc/passwd
```

（18）awk 统计服务器状态连接数，代码如下：

```
netstat -an | awk '/tcp/ {s[$NF]++} END {for(a in s) {print a,s[a]}}'
netstat -an | awk '/tcp/ {print $NF}' | sort | uniq -c
```

17.16　shell 编程四剑客之 grep

全面搜索正则表达式(global search regular expression(re)，grep)是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来。

UNIX/Linux 的 grep 家族包括 grep、egrep 和 fgrep，其中 egrep 和 fgrep 的命令跟 grep

有细微的区别，egrep 是 grep 的扩展，支持更多的 re 元字符，fgrep 是 fixed grep 或 fast grep 的简写，它们把所有的字母都看作单词，正则表达式中的元字符表示其自身的字面意义，不再有其他特殊的含义，一般使用比较少。

目前 Linux 操作系统默认使用 GNU 版本的 grep。它功能更强，可以通过-G、-E、-F 命令行选项来使用 egrep 和 fgrep 的功能。其语法格式如下：

```
grep    -[acinv]    'word'    Filename
```

grep 常用参数详解如下：

- -a：以文本文件方式搜索。
- -c：计算找到的符合行的次数。
- -i：忽略大小写。
- -n：顺便输出行号。
- -v：反向选择，即显示不包含匹配文本的所有行。
- -h：查询多文件时不显示文件名。
- -l：查询多文件时只输出包含匹配字符的文件名。
- -s：不显示不存在或无匹配文本的错误信息。
- -E：允许使用 egrep 扩展模式匹配。

学习 grep 时，需要了解通配符、正则表达式两个概念，很多读者容易把彼此混淆，通配符主要用在 Linux 的 shell 命令中，常用于文件或者文件名称的操作，而正则表达式用于文本内容中的字符串搜索和替换，常用在 awk、grep、sed、vim 工具中对文本的操作。

通配符类型详解如下：

- *：0 个或者多个字符、数字。
- ?：匹配任意一个字符。
- #：表示注解。
- |：管道符号。
- ;：多个命令连续执行。
- &：后台运行指令。
- !：逻辑运算非。
- []：内容范围，匹配括号中内容。
- {}：命令块，多个命令匹配。

正则表达式详解如下：

- *：前一个字符匹配 0 次或多次。
- .：匹配除了换行符以外任意一个字符。
- .*：代表任意字符。
- ^：匹配行首，即以某个字符开头。
- $：匹配行尾，即以某个字符结尾。

- \(..\)：标记匹配字符。
- []：匹配中括号里的任意指定字符，但只匹配一个字符。
- [^]：匹配除中括号以外的任意一个字符。
- \：转义符，取消特殊含义。
- \<：锚定单词的开始。
- \>：锚定单词的结束。
- {n}：匹配字符出现 n 次。
- {n,}：匹配字符出现大于等于 n 次。
- {n,m}：匹配字符至少出现 n 次，最多出现 m 次。
- \w：匹配文字和数字字符。
- \W：\w 的反置形式，匹配一个或多个非单词字符。
- \b：单词锁定符。
- \s：匹配任何空白字符。
- \d：匹配一个数字字符，等价于[0-9]。

常用 grep 工具企业演练案例详解如下：

- grep -c "test"：jfedu.txt 统计 test 字符总行数。
- grep -i "TEST"：jfedu.txt 不区分大小写查找 TEST 所有的行。
- grep -n "test"：jfedu.txt 打印 test 的行及行号。
- grep -v "test"：jfedu.txt 不打印 test 的行。
- grep "test[53]"：jfedu.txt 以字符 test 开头，接 5 或者 3 的行。
- grep "^[^test]"：jfedu.txt 显示输出行首不是 test 的行。
- grep "[Mm]ay"：jfedu.txt 匹配 M 或 m 开头的行。
- grep "K…D"：jfedu.txt 匹配 K，三个任意字符，紧接 D 的行。
- grep "[A-Z][9]D"：jfedu.txt 匹配大写字母，紧跟 9D 的字符行。
- grep "T\{2,\}"：jfedu.txt 打印字符 T 字符连续出现 2 次以上的行。
- grep "T\{4,6\}"：jfedu.txt 打印字符 T 字符连续出现 4 次及 6 次的行。
- grep -n "^$"：jfedu.txt 打印空行的所在的行号。
- grep -vE "#|^$"：jfedu.txt 不匹配文件中的#和空行。
- grep --color -ra -E "db|config|sql" *：jfedu.txt 匹配包含 db 或者 config 或者 sql 的文件。
- grep --color -E "\<([0-9]{1,3}\.){3}([0-9]{1,3})\>"：jfedu.txt 匹配 IPv4 地址。

17.17　shell 数组编程

数组是相同数据类型的元素按一定顺序排列的集合，把有限个类型相同的变量用一个名字命名，然后用编号区分它们变量的集合，这个名称即为数组名，编号即为下标。Linux

shell 编程中常用一维数组。

数组的设计实际上是为了处理方便，把具有相同类型的若干变量按有序的形式组织起来的一种形式，以减少重复频繁的单独定义，如图 17-3 所示。

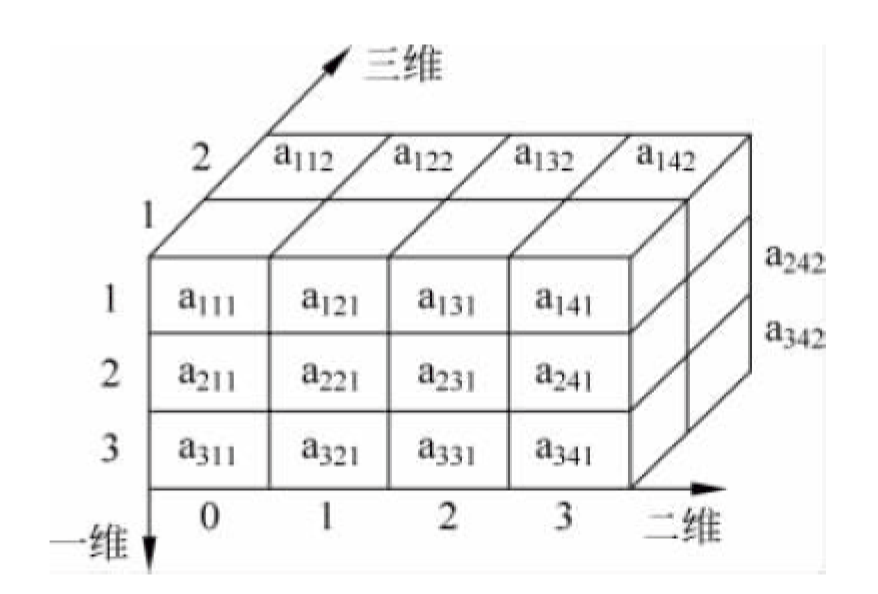


图 17-3　一维、二维、三维数组

定义数组一般以小括号的方式来定义，数组的值可以随机指定，以下为一维数组的定义、统计、引用和删除操作详解。

（1）一维数组定义及创建，代码如下：

```
JFTEST = (
        test1
        test2
        test3
)
LAMP = (httpd php php - devel php - mysql mysql mysql - server)
```

（2）数组下标一般从 0 开始，引用数组的方法详解如下：

- echo ${JFTEST[0]}：引用第一个数组变量，结果打印 test1。
- echo ${JFTEST[1]}：引用第二个数组变量。
- echo ${JFTEST[@]}：显示该数组所有参数。
- echo ${#JFTEST[@]}：显示该数组参数个数。
- echo ${#JFTEST[0]}：显示 test1 字符长度。
- echo ${JFTEST[@]:0}：打印数组所有的值。
- echo ${JFTEST[@]:1}：打印从第二个值开始的所有值。
- echo ${JFTEST[@]:0:2}：打印第一个值与第二个值。
- echo ${JFTEST[@]:1:2}：打印第二个值与第三个值。

（3）数组替换操作，详解如下：

- JFTEST=([0]=www1 [1]=www2 [2]=www3)：数组赋值。
- echo ${JFTEST[@]/test/jfedu}：将数组值 test 替换为 jfedu。
- NEWJFTEST='echo ${JFTEST[@]/test/jfedu}'：将结果赋值新数组。

（4）数组删除操作，详解如下：

- unset array[0]：删除数组第一个值。
- unset array[1]：删除数组第二个值。
- unset array：删除整个数组。

（5）数组 shell 脚本企业案例一，网卡 bond 绑定脚本，代码如下：

```
#!/bin/bash
#Auto Make KVM Virtualization
#Auto config bond scripts
#By author jfedu.net 2017
```

```
eth_bond()
{
NETWORK=(
  HWADDR='ifconfig eth0 |egrep "HWaddr|Bcast" |tr "\n" " "|awk '{print $5,$7,$NF}'|sed -e '
s/addr://g' -e 's/Mask://g'|awk '{print $1}''
  IPADDR='ifconfig eth0 |egrep "HWaddr|Bcast" |tr "\n" " "|awk '{print $5,$7,$NF}'|sed -e '
s/addr://g' -e 's/Mask://g'|awk '{print $2}''
  NETMASK='ifconfig eth0 |egrep "HWaddr|Bcast" |tr "\n" " "|awk '{print $5,$7,$NF}'|sed -e
's/addr://g' -e 's/Mask://g'|awk '{print $3}''
  GATEWAY='route -n|grep "UG"|awk '{print $2}''
)
cat > ifcfg-bond0 << EOF
DEVICE=bond0
BOOTPROTO=static
${NETWORK[1]}
${NETWORK[2]}
${NETWORK[3]}
ONBOOT=yes
TYPE=Ethernet
NM_CONTROLLED=no
EOF
```

(6) 数组 shell 脚本企业案例二，定义 IPv4 值，代码如下：

```
#!/bin/bash
#auto Change ip netmask gateway scripts
#By author jfedu.net 2017
ETHCONF=/etc/sysconfig/network-scripts/ifcfg-eth0
HOSTS=/etc/hosts
NETWORK=/etc/sysconfig/network
DIR=/data/backup/'date +%Y%m%d'
NETMASK=255.255.255.0
echo "----------------------------"
count_ip(){
count=('echo $IPADDR|awk -F. '{print $1,$2,$3,$4}'')
        IP1= ${count[0]}
        IP2= ${count[1]}
        IP3= ${count[2]}
        IP4= ${count[3]}
}
```

第 18 章



shell 编程高级企业实战

企业生产环境中，服务器规模成百上千，如果依靠人工去维护和管理是非常吃力的，基于 shell 编程脚本管理和维护服务器变得简单、从容，而且对企业自动化运维之路的建设起到极大的推动作用。

本章向读者介绍企业生产环境 shell 编程案例、自动化备份 MySQL 数据、服务器信息收集、防止恶意 IP 访问、LAMP＋MySQL 主从实战、千台服务器 IP 修改、Nginx＋Tomcat 高级自动化部署脚本、Nginx 虚拟主机配置、Docker 管理平台等内容。

18.1　shell 编程实战系统备份脚本

日常企业运维中，需要备份 Linux 操作系统中重要的文件和目录，例如/etc、/boot 分区、重要网站数据等，在备份数据时，由于数据量非常大，需要指定高效的备份方案，以下为常用的备份数据方案：

- 每周日进行完整备份，周一至周六使用增量备份；
- 每周六进行完整备份，周日至周五使用增量备份。

企业备份数据的工具主要有 tar、cp、rsync、scp、sersync、dd 等工具。以下为基于开源 tar 工具实现系统数据备份方案。

tar 工具手动全备份网站，-g 参数指定新的快照文件，代码如下：

```
tar -g   /tmp/snapshot   -czvf   /tmp/2017_full_system_data.tar.gz /data/sh/
```

tar 工具手动增量备份网站，-g 参数指定全备已生成的快照文件，后续增量备份基于上一个增量备份快照文件，代码如下：

```
tar -g   /tmp/snapshot   -czvf   /tmp/2014_add01_system_data.tar.gz /data/sh/
```

tar 工具全备、增量备份网站，shell 脚本实现自动打包备份，编程思路如下：

- 系统备份数据按每天存放；
- 创建完整备份函数块；
- 创建增量备份函数块；

- 根据星期数判断完整或增量；
- 将脚本加入 crontab 实现自动备份。

tar 工具全备、增量备份网站，shell 脚本实现自动打包备份，代码如下：

```
#!/bin/bash
#Auto Backup Linux System Files
#By author jfedu.net 2017
#Define Path variables
SOURCE_DIR=(
    $*
)
TARGET_DIR=/data/backup/
YEAR='date +%Y'
MONTH='date +%m'
DAY='date +%d'
WEEK='date +%u'
A_NAME='date +%H%M'
FILES=system_backup.tgz
CODE=$?
if
    [ -z "$*" ]; then
    echo -e "\033[32mUsage:\nPlease Enter Your Backup Files or Directories\n------------
----------------------------------\n\nUsage: { $0 /boot /etc}\033[0m"
    exit
fi
#Determine Whether the Target Directory Exists
if
    [ ! -d $TARGET_DIR/$YEAR/$MONTH/$DAY ]; then
    mkdir -p $TARGET_DIR/$YEAR/$MONTH/$DAY
    echo -e "\033[32mThe $TARGET_DIR Created Successfully !\033[0m"
fi
#EXEC Full_Backup Function Command
Full_Backup()
{
if
    [ "$WEEK" -eq "7" ]; then
    rm -rf $TARGET_DIR/snapshot
    cd $TARGET_DIR/$YEAR/$MONTH/$DAY ; tar -g $TARGET_DIR/snapshot -czvf $FILES ${SOURCE_
DIR[@]}
    [ "$CODE" == "0" ]&&echo -e "-----------------------------------------------
---\n\033[32mThese Full_Backup System Files Backup Successfully !\033[0m"
fi
}
#Perform incremental BACKUP Function Command
Add_Backup()
{
```

```
    if
        [ $WEEK -ne "7" ]; then
        cd $TARGET_DIR/$YEAR/$MONTH/$DAY ; tar -g $TARGET_DIR/snapshot -czvf $A_NAME
$FILES ${SOURCE_DIR[@]}
        [ "$CODE" == "0" ]&&echo -e "----------------------------------------
---\n\033[32mThese Add_Backup System Files $TARGET_DIR/$YEAR/$MONTH/$DAY/${YEAR}_$A_NAME
$FILES Backup Successfully!\033[0m"
    fi
}
sleep 3
Full_Backup; Add_Backup
```

crontab 任务计划中添加如下语句，每天凌晨 1 点整执行备份脚本即可：

```
0 1 * * * /bin/sh /data/sh/auto_backup.sh /boot /etc/ >> /tmp/back.log 2>&1
```

18.2 shell 编程实战收集服务器信息

在企业生产环境中，经常会对服务器资产进行统计存档，单台服务器可以手动去统计服务器的 CPU 型号、内存大小、硬盘容量、网卡流量等，但如果服务器数量超过百台、千台，使用手工方式就变得非常吃力。

基于 shell 脚本实现自动化服务器硬件信息的收集，并将收集的内容存放在数据库，能更快、更高效地实现对服务器资产信息的管理。shell 脚本实现服务器信息自动收集，编程思路如下：

- 创建数据库和表存储服务器信息；
- 基于 shell 四剑客 awk、find、sed、grep 获取服务器信息；
- 将获取的信息写成 SQL 语句；
- 定期对 SQL 数据进行备份；
- 将脚本加入 crontab 实现自动备份。

创建数据库表，创建 SQL 语句，代码如下：

```
CREATE TABLE 'audit_system' (
  'id' int(11) NOT NULL AUTO_INCREMENT,
  'ip_info' varchar(50) NOT NULL,
  'serv_info' varchar(50) NOT NULL,
  'cpu_info' varchar(50) NOT NULL,
  'disk_info' varchar(50) NOT NULL,
  'mem_info' varchar(50) NOT NULL,
  'load_info' varchar(50) NOT NULL,
  'mark_info' varchar(50) NOT NULL,
  PRIMARY KEY ('id'),
  UNIQUE KEY 'ip_info' ('ip_info'),
```

```
UNIQUE KEY 'ip_info_2' ('ip_info')
);
```

shell 脚本实现服务器信息自动收集，代码如下：

```
#!/bin/bash
#Auto get system info
#By author jfedu.net 2017
#Define Path variables
echo -e "\033[34m \033[1m"
cat << EOF
++++++++++++++++++++++++++++++++++++++++++++++
+++++++Welcome to use system Collect+++++++ +
++++++++++++++++++++++++++++++++++++++++++++++
EOF
ip_info = 'ifconfig |grep "Bcast"|tail -1 |awk '{print $2}'|cut -d: -f 2'
cpu_info1 = 'cat /proc/cpuinfo |grep 'model name'|tail -1 |awk -F: '{print $2}'|sed 's/^//g'
|awk '{print $1, $3, $4, $NF}''
cpu_info2 = 'cat /proc/cpuinfo |grep "physical id"|sort |uniq -c|wc -l'
serv_info = 'hostname |tail -1'
disk_info = 'fdisk -l|grep "Disk"|grep -v "identifier"|awk '{print $2, $3, $4}'|sed 's/,//g''
mem_info = 'free -m |grep "Mem"|awk '{print "Total", $1, $2"M"}''
load_info = 'uptime |awk '{print "Current Load: " $(NF-2)}'|sed 's/\,//g''
mark_info = 'BeiJing_IDC'
echo -e "\033[32m-----------------------------------------------\033[1m"
echo IPADDR: ${ip_info}
echo HOSTNAME: $serv_info
echo CPU_INFO: ${cpu_info1} X ${cpu_info2}
echo DISK_INFO: $disk_info
echo MEM_INFO: $mem_info
echo LOAD_INFO: $load_info
echo -e "\033[32m-----------------------------------------------\033[0m"
echo -e -n "\033[36mYou want to write the data to the databases? \033[1m" ; read ensure
if[ "$ensure" == "yes" -o "$ensure" == "y" -o "$ensure" == "Y" ]; then
    echo "--------------------------------------------"
    echo -e '\033[31mmysql -uaudit -p123456 -D audit -e ''' "insert into audit_system
values('','${ip_info}','$serv_info','${cpu_info1} X ${cpu_info2}','$disk_info','$mem_info
','$load_info','$mark_info')" ''' \033[0m '
    mysql -uroot -p123456 -D test -e "insert into audit_system values('','${ip_info}',
'$serv_info','${cpu_info1} X ${cpu_info2}','$disk_info','$mem_info','$load_info','$mark_
info')"
else
    echo "Please wait,exit......"
    exit
fi
```

手动读取数据库服务器信息命令，代码如下：

```
mysql -uroot -p123 -e 'use wugk1 ; select * from audit_audit_system; '|sed 's/-//g'|grep -v "id"
```

18.3 shell 编程实战拒绝恶意 IP 登录

企业服务器暴露在外网，每天会有大量的人使用各种用户名和密码尝试登录服务器，如果让其一直尝试，难免会猜出密码，通过开发 shell 脚本，可以自动将尝试登录服务器错误密码超过设定次数的 IP 列表加入到防火墙配置中。

shell 脚本实现服务器拒绝恶意 IP 登录，编程思路如下：

- 登录服务器日志/var/log/secure；
- 检查日志中认证失败的行并打印其 IP 地址；
- 将 IP 地址写入至防火墙；
- 禁止该 IP 访问服务器 SSH 22 端口；
- 将脚本加入 crontab 实现自动禁止恶意 IP。

shell 脚本实现服务器拒绝恶意 IP 登录，代码如下：

```
#!/bin/bash
#Auto drop ssh failed IP address
#By author jfedu.net 2017
#Define Path variables
SEC_FILE=/var/log/secure
IP_ADDR='awk '{print $0}' /var/log/secure|grep -i "fail"| egrep -o "([0-9]{1,3}\.){3}
[0-9]{1,3}" | sort -nr | uniq -c |awk '$1>=15 {print $2}''
IPTABLE_CONF=/etc/sysconfig/iptables
echo
cat << EOF
++++++++++++++welcome to use ssh login drop failed ip+++++++++++++++ +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++ ------------------------------------- ++++++++++++++++++
EOF
echo
for ((j=0; j<=6; j++)) ; do echo -n "-"; sleep 1 ; done
echo
for i in 'echo $IP_ADDR'
do
    cat $IPTABLE_CONF |grep $i >/dev/null
if
    [ $? -ne 0 ]; then
    sed -i "/lo/a -A INPUT -s $i -m state --state NEW -m tcp -p tcp --dport 22 -j
DROP" $IPTABLE_CONF
fi
done
NUM='find /etc/sysconfig/ -name iptables -a -mmin -1|wc -l'
        if [ $NUM -eq 1 ]; then
```

```
            /etc/init.d/iptables restart
        fi
```

18.4 shell 编程实战 LAMP 一键安装

LAMP 是目前互联网主流 Web 网站架构，通过源码安装、维护和管理对于单台很轻松，但如果服务器数量多，手工管理就非常困难，基于 shell 脚本可以更快速地维护 LAMP 架构。

shell 脚本实现服务器 LAMP 一键源码安装配置，编程思路如下：

- 脚本的功能，实现安装 LAMP 环境、论坛网站；
- Apache 安装配置、MySQL、PHP 安装；
- 源码 LAMP 整合配置；
- 启动数据库，创建数据库并授权；
- 重启 LAMP 所有服务，验证访问。

shell 脚本实现服务器 LAMP 一键源码安装配置，代码如下：

```
#!/bin/bash
#Auto install LAMP
#By author jfedu.net 2017
#Define Path variables
#Httpd define path variable
H_FILES=httpd-2.2.32.tar.bz2
H_FILES_DIR=httpd-2.2.32
H_URL=http://mirrors.cnnic.cn/apache/httpd/
H_PREFIX=/usr/local/apache2/
#MySQL define path variable
M_FILES=mysql-5.5.20.tar.gz
M_FILES_DIR=mysql-5.5.20
M_URL=http://down1.chinaunix.net/distfiles/
M_PREFIX=/usr/local/mysql/
#PHP define path variable
P_FILES=php-5.3.28.tar.bz2
P_FILES_DIR=php-5.3.28
P_URL=http://mirrors.sohu.com/php/
P_PREFIX=/usr/local/php5/
function httpd_install(){
if [[ "$1" -eq "1" ]]; then
    wget -c $H_URL/$H_FILES && tar -jxvf $H_FILES && cd $H_FILES_DIR &&./configure --
prefix=$H_PREFIX
    if [ $? -eq 0 ]; then
        make && make install
    fi
fi
```

```
}
function mysql_install(){
if [[ "$1" -eq "2" ]]; then
wget -c $M_URL/$M_FILES && tar -xzvf $M_FILES && cd $M_FILES_DIR &&yum install cmake
ncurses-devel -y ; cmake . -DCMAKE_INSTALL_PREFIX=$M_PREFIX \
-DMYSQL_UNIX_ADDR=/tmp/mysql.sock \
-DMYSQL_DATADIR=/data/mysql \
-DSYSCONFDIR=/etc \
-DMYSQL_USER=mysql \
-DMYSQL_TCP_PORT=3306 \
-DWITH_XTRADB_STORAGE_ENGINE=1 \
-DWITH_INNOBASE_STORAGE_ENGINE=1 \
-DWITH_PARTITION_STORAGE_ENGINE=1 \
-DWITH_BLACKHOLE_STORAGE_ENGINE=1 \
-DWITH_MYISAM_STORAGE_ENGINE=1 \
-DWITH_READLINE=1 \
-DENABLED_LOCAL_INFILE=1 \
-DWITH_EXTRA_CHARSETS=1 \
-DDEFAULT_CHARSET=utf8 \
-DDEFAULT_COLLATION=utf8_general_ci \
-DEXTRA_CHARSETS=all \
-DWITH_BIG_TABLES=1 \
-DWITH_DEBUG=0
if [ $? -eq 0 ]; then
    make && make install
    echo -e "\n\033[32m-----------------------------------------------\033[0m"
          echo -e "\033[32mThe $M_FILES_DIR Server Install Success !\033[0m"
        else
          echo -e "\033[32mThe $M_FILES_DIR Make or Make install ERROR,Please Check......"
          exit 0
fi
/bin/cp support-files/my-small.cnf /etc/my.cnf
/bin/cp support-files/mysql.server /etc/init.d/mysqld
chmod +x /etc/init.d/mysqld
chkconfig --add mysqld
chkconfig mysqld on
fi
}
function php_install(){
if [[ "$1" -eq "3" ]]; then
        yum install libxml2-devel perl-devel perl libtool* -y
        wget -c $P_URL/$P_FILES && tar -jxvf $P_FILES && cd $P_FILES_DIR &&./configure --
prefix=$P_PREFIX --with-config-file-path=$P_PREFIX/etc --with-mysql=$M_PREFIX --
with-apxs2=$H_PREFIX/bin/apxs
if [ $? -eq 0 ]; then
          make ZEND_EXTRA_LIBS='-liconv' && make install
          echo -e "\n\033[32m-----------------------------------------------
```

```
-\033[0m"
          echo -e "\033[32mThe $P_FILES_DIR Server Install Success !\033[0m"
        else
          echo -e "\033[32mThe $P_FILES_DIR Make or Make install ERROR,Please Check......"
          exit 0
        fi
fi
}
function lamp_config(){
if [[ "$1" -eq "4" ]]; then
    sed -i '/DirectoryIndex/s/index.html/index.php index.html/g' $H_PREFIX/conf/httpd.conf
     $H_PREFIX/bin/apachectl restart
    echo "AddType      application/x-httpd-php .php" >> $H_PREFIX/conf/httpd.conf
    IP='ifconfig eth0|grep "Bcast"|awk '{print $2}'|cut -d: -f2'
    echo "You can to access http://$IP/"

cat >$H_PREFIX/htdocs/index.php <<EOF
<?php
phpinfo();
?>
EOF
fi
}
PS3="Please enter you select install menu:"
select i in http mysql php config quit
do

case $i in
    http)
    httpd_install 1
    ;;
    mysql)
    mysql_install 2
    ;;
    php)
    php_install 3
    ;;
    config)
    lamp_config 4
    ;;
    quit)
    exit
esac
done
```

18.5 shell 编程实战 MySQL 主从复制

MySQL 数据库服务器主要应用于与动态网站结合，存放网站必要的数据，例如订单、交易、员工表、薪资等记录，为了实现数据备份，需引入 MySQL 主从架构，MySQL 主从架构脚本可以实现自动化安装、配置和管理。

shell 脚本实现服务器 MySQL 一键 YUM 安装配置，编程思路如下：

(1) MySQL 主库的操作：

- 主库上安装 MySQL，设置 server-id、bin-log；
- 授权复制同步的用户，对客户端授权；
- 确认 bin-log 文件名、position 位置点。

(2) MySQL 从库的操作：

- 从库上安装 MySQL，设置 server-id；
- change master 指定主库和 bin-log 名以及 position；
- start slave 启动从库 I/O 线程；
- show slave status\G 查看主从的状态。

shell 脚本实现服务器 MySQL 一键 YUM 安装配置，需要提前手动授权主库可以免密码登录从库服务器，代码如下：

```
#!/bin/bash
#Auto install Mysql AB Repliation
#By author jfedu.net 2017
#Define Path variables
MYSQL_SOFT="mysql mysql-server mysql-devel php-mysql mysql-libs"
NUM='rpm -qa |grep -i mysql |wc -l'
INIT="/etc/init.d/mysqld"
CODE=$?
#Mysql To Install 2017
if [ $NUM -ne 0 -a -f $INIT ]; then
    echo -e "\033[32mThis Server Mysql already Install.\033[0m"
    read -p "Please ensure yum remove Mysql Server,YES or NO": INPUT
    if [ $INPUT == "y" -o $INPUT == "yes" ]; then
        yum remove $MYSQL_SOFT -y ; rm -rf /var/lib/mysql /etc/my.cnf
        yum install $MYSQL_SOFT -y
    else
        echo
    fi
else
    yum remove $MYSQL_SOFT -y ; rm -rf /var/lib/mysql /etc/my.cnf
    yum install $MYSQL_SOFT -y
    if [ $CODE -eq 0 ]; then
        echo -e "\033[32mThe Mysql Install Successfully.\033[0m"
```

```
    else
        echo -e "\033[32mThe Mysql Install Failed.\033[0m"
        exit 1
    fi
fi
my_config(){
cat >/etc/my.cnf << EOF
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
symbolic-links=0
log-bin=mysql-bin
server-id = 1
auto_increment_offset=1
auto_increment_increment=2
[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
EOF
}
my_config
/etc/init.d/mysqld restart
ps -ef |grep mysql
MYSQL_CONFIG(){
#Master Config Mysql
mysql -e "grant replication slave on *.* to 'tongbu'@'%' identified by '123456'; "
MASTER_FILE='mysql -e "show master status; "|tail -1|awk '{print $1}''
MASTER_POS='mysql -e "show master status; "|tail -1|awk '{print $2}''
MASTER_IPADDR='ifconfig eth0|grep "Bcast"|awk '{print $2}'|cut -d: -f2'
read -p "Please Input Slave IPaddr: " SLAVE_IPADDR
#Slave Config Mysql
ssh -l root $SLAVE_IPADDR "yum remove $MYSQL_SOFT -y ; rm -rf /var/lib/mysql /etc/my.cnf ;
yum install $MYSQL_SOFT -y"
ssh -l root $SLAVE_IPADDR " $my_config"
#scp -r /etc/my.cnf root@192.168.111.129:/etc/
ssh -l root $SLAVE_IPADDR "sed -i 's#server-id = 1#server-id = 2#g' /etc/my.cnf"
ssh -l root $SLAVE_IPADDR "sed -i '/log-bin=mysql-bin/d' /etc/my.cnf"
ssh -l root $SLAVE_IPADDR "/etc/init.d/mysqld restart"
ssh -l root $SLAVE_IPADDR "mysql -e \"change master to master_host='$MASTER_IPADDR',master
_user='tongbu',master_password='123456',master_log_file='$MASTER_FILE',master_log_pos=
 $MASTER_POS; \""
ssh -l root $SLAVE_IPADDR "mysql -e \"slave start; \""
ssh -l root $SLAVE_IPADDR "mysql -e \"show slave status\G; \""
}

read -p "Please ensure your Server is Master and you will config mysql Replication? yes or
```

```
no": INPUT
if [ $INPUT == "y" -o $INPUT == "yes" ]; then
    MYSQL_CONFIG
else
    exit 0
fi
```

18.6 shell 编程实战修改 IP 及主机名

企业中服务器 IP 地址系统通过自动化工具安装完系统，IP 均是自动获取的，而服务器要求固定的静态 IP，百台服务器手工去配置静态 IP 是不可取的，可以基于 shell 脚本自动修改 IP、主机名等信息。

shell 脚本实现服务器 IP、主机名自动修改及配置，编程思路如下：

- 静态 IP 修改；
- 动态 IP 修改；
- 根据 IP 生成主机名并配置；
- 修改 DNS 域名解析。

shell 脚本实现服务器 IP、主机名自动修改及配置，代码如下：

```
#!/bin/bash
#Auto Change ip netmask gateway scripts
#By author jfedu.net 2017
#Define Path variables
ETHCONF=/etc/sysconfig/network-scripts/ifcfg-eth0
HOSTS=/etc/hosts
NETWORK=/etc/sysconfig/network
DIR=/data/backup/'date +%Y%m%d'
NETMASK=255.255.255.0
echo "------------------------------"
judge_ip(){
    read -p "Please enter ip Address,example 192.168.0.11 ip": IPADDR
    echo $IPADDR|grep -v "[Aa-Zz]"|grep --color -E "([0-9]{1,3}\.){3}[0-9]{1,3}"
}
count_ip(){
    count=('echo $IPADDR|awk -F. '{print $1, $2, $3, $4}'')
    IP1= ${count[0]}
    IP2= ${count[1]}
    IP3= ${count[2]}
    IP4= ${count[3]}
}
ip_check()
{
judge_ip
```

```
while [ $? -ne 0 ]
do
    judge_ip
done
count_ip
while [ "$IP1" -lt 0 -o "$IP1" -ge 255 -o "$IP2" -ge 255 -o "$IP3" -ge 255 -o "$IP4"
 -ge 255 ]
do
    judge_ip
    while [ $? -ne 0 ]
    do
        judge_ip
    done
    count_ip
done
}
change_ip()
{
if [ ! -d $DIR ]; then
    mkdir -p $DIR
fi
echo "The Change ip address to Backup Interface eth0"
cp $ETHCONF $DIR
grep "dhcp" $ETHCONF
if [ $? -eq 0 ]; then
    read -p "Please enter ip Address:" IPADDR
    sed -i 's/dhcp/static/g' $ETHCONF
    echo -e "IPADDR= $IPADDR\nNETMASK= $NETMASK\nGATEWAY= 'echo $IPADDR|awk -F. '{print
 $1"."$2"."$3}''.2" >> $ETHCONF
    echo "The IP configuration success. !"
else
    echo -n "Static IP has been configured,please confirm whether to modify,yes or No":
    read i
fi
if [ "$i" == "y" -o "$i" == "yes" ]; then
    ip_check
    sed -i -e '/IPADDR/d' -e '/NETMASK/d' -e '/GATEWAY/d' $ETHCONF
    echo -e "IPADDR= $IPADDR\nNETMASK= $NETMASK\nGATEWAY= 'echo $IPADDR|awk -F. '{print
 $1"."$2"."$3}''.2" >> $ETHCONF
    echo "The IP configuration success. !"
    echo
else
    echo "Static IP already exists,please exit."
    exit $?
fi
}
change_hosts()
```

```
{

if [ ! -d $DIR ]; then
    mkdir -p $DIR
fi
cp $HOSTS $DIR
ip_check
host = 'echo $IPADDR|sed 's/\./-/g'|awk '{print "BJ-IDC-" $0"-jfedu.net"}''
cat $HOSTS |grep "$host"
if [ $? -ne 0 ]; then
    echo "$IPADDR    $host" >> $HOSTS
    echo "The hosts modify success "
fi
grep "$host" $NETWORK
if [ $? -ne 0 ]; then
    sed -i "s/^HOSTNAME/#HOSTNAME/g" $NETWORK
    echo "NETWORK= $host" >> $NETWORK
    hostname $host; su
fi
}
PS3 = "Please Select configuration ip or configuration host:"
select i in "modify_ip" "modify_hosts" "exit"
do
    case $i in
            modify_ip)
            change_ip
        ;;
            modify_hosts)
            change_hosts
        ;;
            exit)
            exit
        ;;
            *)
            echo -e "1) modify_ip\n2) modify_ip\n3)exit"
    esac

done
```

18.7 shell 编程实战 Zabbix 安装配置

Zabbix 是一款分布式监控系统，基于 C/S 模式，需在服务器安装 zabbix_server，在客户端安装 zabbix_agent，通过 shell 脚本可以更快速地实现该需求。

shell 脚本实现 Zabbix 服务器端和客户端自动安装，编程思路如下：

- Zabbix 软件的版本源码安装、路径、--enable-server、--enable-agent；
- cp zabbix_agentd 启动进程-/etc/init.d/zabbix_agentd，执行 x 权限；
- 配置 zabbix_agentd.conf 文件，指定 server IP 变量；
- 指定客户端的 Hostname，可以写成客户端 IP 地址；
- 启动 zabbix_agentd 服务，创建 Zabbix user。

shell 脚本实现 Zabbix 服务器端和客户端自动安装，代码如下：

```
#!/bin/bash
#Auto install zabbix server and client
#By author jfedu.net 2017
#Define Path variables
ZABBIX_SOFT="zabbix-3.2.6.tar.gz"
INSTALL_DIR="/usr/local/zabbix/"
SERVER_IP="192.168.111.128"
IP='ifconfig|grep Bcast|awk '{print $2}'|sed 's/addr://g''
SERVER_INSTALL(){
yum -y install curl curl-devel net-snmp net-snmp-devel perl-DBI
groupadd zabbix ; useradd -g zabbix zabbix; usermod -s /sbin/nologin zabbix
tar -xzf $ZABBIX_SOFT; cd 'echo $ZABBIX_SOFT|sed 's/.tar.*//g''
./configure --prefix=/usr/local/zabbix --enable-server --enable-agent --with-mysql --
enable-ipv6 --with-net-snmp --with-libcurl &&make install
if [ $? -eq 0 ]; then
    ln -s /usr/local/zabbix/sbin/zabbix_* /usr/local/sbin/
fi
cd - ; cd zabbix-3.2.6
cp misc/init.d/tru64/{zabbix_agentd,zabbix_server} /etc/init.d/ ; chmod o+x /etc/init.d/
zabbix_*
mkdir -p /var/www/html/zabbix/; cp -a frontends/php/* /var/www/html/zabbix/
#config zabbix server
cat >$INSTALL_DIR/etc/zabbix_server.conf << EOF
LogFile=/tmp/zabbix_server.log
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=123456
EOF
#config zabbix agentd
cat >$INSTALL_DIR/etc/zabbix_agentd.conf << EOF
LogFile=/tmp/zabbix_agentd.log
Server= $SERVER_IP
ServerActive= $SERVER_IP
Hostname = $IP
EOF
#start zabbix agentd
/etc/init.d/zabbix_server restart
```

```
/etc/init.d/zabbix_agentd restart
/etc/init.d/iptables stop
setenforce 0
}
AGENT_INSTALL(){
yum -y install curl curl-devel net-snmp net-snmp-devel perl-DBI
groupadd zabbix ; useradd -g zabbix zabbix; usermod -s /sbin/nologin zabbix

tar -xzf $ZABBIX_SOFT; cd 'echo $ZABBIX_SOFT|sed 's/.tar.*//g''
./configure --prefix=/usr/local/zabbix --enable-agent&&make install
if [ $? -eq 0 ]; then
    ln -s /usr/local/zabbix/sbin/zabbix_* /usr/local/sbin/
fi
cd - ; cd zabbix-3.2.6
cp misc/init.d/tru64/zabbix_agentd /etc/init.d/zabbix_agentd ; chmod o+x /etc/init.d/zabbix_
agentd
#config zabbix agentd
cat >$INSTALL_DIR/etc/zabbix_agentd.conf << EOF
LogFile=/tmp/zabbix_agentd.log
Server= $SERVER_IP
ServerActive= $SERVER_IP
Hostname = $IP
EOF
#start zabbix agentd
/etc/init.d/zabbix_agentd restart
/etc/init.d/iptables stop
setenforce 0
}

read -p "Please confirm whether to install Zabbix Server,yes or no? " INPUT
if [ $INPUT == "yes" -o $INPUT == "y" ]; then
    SERVER_INSTALL
else
    AGENT_INSTALL
fi
```

18.8 shell 编程实战 Nginx 虚拟主机

Nginx Web 服务器的最大特点在于 Nginx 常被用于负载均衡、反向代理，单台 Nginx 服务器配置多个虚拟主机，百台服务器配置 N 个虚拟主机，基于 shell 脚本可以更加高效地配置虚拟主机及添加、管理。

shell 脚本实现 Nginx 自动安装及虚拟主机的维护，编程思路如下：

- 脚本指定参数 v1.jfedu.net；
- 创建 v1.jfedu.net 同时创建目录/var/www/v1；

- 将 Nginx 虚拟主机配置定向到新的目录;
- 重复虚拟主机不再添加。

shell 脚本实现 Nginx 自动安装及虚拟主机的配置,代码如下:

```
#!/bin/bash
#Auto config Nginx virtual Hosts
#By author jfedu.net 2017
#Define Path variables
NGINX_CONF="/usr/local/nginx/conf/"
NGINX_MAKE=" --user=www --group=www --prefix=/usr/local/nginx --with-http_stub_
status_module --with-http_ssl_module"
NGINX_SBIN="/usr/local/nginx/sbin/nginx"
NGINX_INSTALL(){
#Install Nginx server
NGINX_FILE=nginx-1.12.0.tar.gz
NGINX_DIR='echo $NGINX_FILE|sed 's/.tar*.*//g''
if [ ! -e /usr/local/nginx/ -a ! -e /etc/nginx/ ]; then
    pkill nginx
    wget -c http://nginx.org/download/$NGINX_FILE
    yum install pcre-devel pcre -y
    rm -rf $NGINX_DIR ; tar xf $NGINX_FILE
    cd $NGINX_DIR; useradd www; ./configure $NGINX_MAKE
    make &&make install
    grep -vE "#|^$" $NGINX_CONF/nginx.conf >$NGINX_CONF/nginx.conf.swp
    \mv $NGINX_CONF/nginx.conf.swp $NGINX_CONF/nginx.conf
    for i in 'seq 1 6'; do sed -i '$d' $NGINX_CONF/nginx.conf; done
    echo "}" >> $NGINX_CONF/nginx.conf
    cd ../
fi
}
NGINX_CONFIG(){
#config tomcat nginx vhosts
grep "include domains" $NGINX_CONF/nginx.conf >>/dev/null
if [ $? -ne 0 ]; then
    #sed -i '$d' $NGINX_CONF/nginx.conf
    echo -e "\ninclude domains/*; \n}" >> $NGINX_CONF/nginx.conf
    mkdir -p $NGINX_CONF/domains/
fi
VHOSTS=$1
ls $NGINX_CONF/domains/$VHOSTS >>/dev/null 2>&1
if [ $? -ne 0 ]; then
    #cp -r xxx.jfedu.net $NGINX_CONF/domains/$VHOSTS
    #sed -i "s/xxx/$VHOSTS/g" $NGINX_CONF/domains/$VHOSTS
    cat >$NGINX_CONF/domains/$VHOSTS << EOF
    #vhost server $VHOSTS
    server {
```

```
            listen       80;
            server_name  $VHOSTS;
            location / {
                root    /data/www/$VHOSTS/;
                index index.html index.htm;
            }
        }
EOF
    mkdir -p /data/www/$VHOSTS/
    cat >/data/www/$VHOSTS/index.html << EOF
    <html>
    <h1><center>The First Test Nginx page.</center></h1>
    <hr color="red">
    <h2><center>$VHOSTS </center></h2>
    </html>
EOF
    echo -e "\033[32mThe $VHOSTS Config success, You can to access http://$VHOSTS/\033[0m"
    NUM='ps -ef |grep nginx|grep -v grep|grep -v auto|wc -l'
    $NGINX_SBIN -t >>/dev/null 2>&1
    if [ $? -eq 0 -a $NUM -eq 0 ]; then
        $NGINX_SBIN
    else
        $NGINX_SBIN -t >>/dev/null 2>&1
        if [ $? -eq 0 ]; then
            $NGINX_SBIN -s reload
        fi
    fi
else
    echo -e "\033[32mThe $VHOSTS has been config,Please exit.\033[0m"
fi
}
if [ -z $1 ]; then
    echo -e "\033[32m--------------------\033[0m"
    echo -e "\033[32mPlease enter sh $0 xx.jf.com.\033[0m"
    exit 0
fi
NGINX_INSTALL
NGINX_CONFIG $1
```

18.9 shell 编程实战 Nginx、Tomcat 脚本

Tomcat 用于发布 JSP Web 页面，根据企业实际需求，会在单台服务器配置 N 个 Tomcat 实例，同时手动将 Tomcat 创建后的实例加入至 Nginx 虚拟主机中，同时重启 Nginx，开发 Nginx、Tomcat 自动创建 Tomcat 实例及 Nginx 虚拟主机管理脚本，这能大大

地减轻人工干预，实现快速地交付。

shell 脚本实现 Nginx 自动安装、虚拟主机及自动将 Tomcat 加入至虚拟主机，编程思路如下：

- 手动复制 Tomcat 与脚本一致目录（可自动修改）；
- 手动修改 Tomcat 端口为 6001、7001、8001（可自动修改）；
- 脚本指定参数 v1.jfedu.net；
- 创建 v1.jfedu.net Tomcat 实例；
- 修改 Tomcat 实例端口，保证 port 唯一；
- 将 Tomcat 实例加入 Nginx 虚拟主机；
- 重复创建 Tomcat 实例，端口自动增加，并加入原 Nginx 虚拟主机，实现负载均衡。

shell 脚本实现 Nginx 自动安装、虚拟主机及自动将 Tomcat 加入至虚拟主机，代码如下：

```
#!/bin/bash
#Auto config Nginx and tomcat cluster
#By author jfedu.net 2017
#Define Path variables
NGINX_CONF="/usr/local/nginx/conf/"
install_nginx(){
    NGINX_FILE=nginx-1.10.2.tar.gz
    NGINX_DIR='echo $NGINX_FILE|sed 's/.tar*.*//g''
    wget -c http://nginx.org/download/$NGINX_FILE
    yum install pcre-devel pcre -y
    rm -rf $NGINX_DIR ; tar xf $NGINX_FILE
    cd $NGINX_DIR; useradd www; ./configure --user=www --group=www --prefix=/usr/
local/nginx2 --with-http_stub_status_module --with-http_ssl_module
    make &&make install
    cd ../
}
install_tomcat(){
    JDK_FILE="jdk1.7.0_25.tar.gz"
    JDK_DIR='echo $JDK_FILE|sed 's/.tar.*//g''
    tar -xzf $JDK_FILE ; mkdir -p /usr/java/ ; mv $JDK_DIR /usr/java/
    sed -i '/JAVA_HOME/d; /JAVA_BIN/d; /JAVA_OPTS/d' /etc/profile
    cat >> /etc/profile << EOF
    export JAVA_HOME=/export/servers/$JAVA_DIR
    export JAVA_BIN=/export/servers/$JAVA_DIR/bin
    export PATH=\$JAVA_HOME/bin:\$PATH
    export CLASSPATH=.:\$JAVA_HOME/lib/dt.jar:\$JAVA_HOME/lib/tools.jar
    export JAVA_HOME JAVA_BIN PATH CLASSPATH
EOF
    source /etc/profile; java -version
    #install tomcat start
```

```
    ls tomcat
}
config_tomcat_nginx(){
    #config tomcat nginx vhosts
    grep "include domains" $NGINX_CONF/nginx.conf >>/dev/null
    if [ $? -ne 0 ]; then
        sed -i '$d' $NGINX_CONF/nginx.conf
        echo -e "\ninclude domains/*; \n}" >> $NGINX_CONF/nginx.conf
        mkdir -p $NGINX_CONF/domains/
    fi
    VHOSTS=$1
    NUM='ls /usr/local/|grep -c tomcat'
    if [ $NUM -eq 0 ]; then
        cp -r tomcat /usr/local/tomcat_$VHOSTS
        cp -r xxx.jfedu.net $NGINX_CONF/domains/$VHOSTS
        #sed -i "s/VHOSTS/$VHOSTS/g" $NGINX_CONF/domains/$VHOSTS
        sed -i "s/xxx/$VHOSTS/g" $NGINX_CONF/domains/$VHOSTS
        exit 0
    fi
    #-------------------------------
    #VHOSTS=$1
    VHOSTS_NUM='ls $NGINX_CONF/domains/|grep -c $VHOSTS'
    SERVER_NUM='grep -c "127" $NGINX_CONF/domains/$VHOSTS'
    SERVER_NUM_1='expr $SERVER_NUM + 1'
    rm -rf /tmp/.port.txt
    for i in 'find /usr/local/ -maxdepth 1 -name "tomcat*"'; do
        grep "port" $i/conf/server.xml |egrep -v "\--|8080|SSLEnabled"|awk '{print $2}'|
sed 's/port=//g; s/\"//g'|sort -nr >>/tmp/.port.txt
    done
    MAX_PORT='cat /tmp/.port.txt|grep -v 8443|sort -nr|head -1'
    PORT_1='expr $MAX_PORT - 2000 + 1'
    PORT_2='expr $MAX_PORT - 1000 + 1'
    PORT_3='expr $MAX_PORT + 1'
    if [ $VHOSTS_NUM -eq 1 ]; then
        read -p "The $VHOSTS is exists,You sure create mulit Tomcat for the $VHOSTS? yes or no
" INPUT
        if [ $INPUT == "YES" -o $INPUT == "Y" -o $INPUT == "yes" ]; then
            cp -r tomcat /usr/local/tomcat_${VHOSTS}_${SERVER_NUM_1}
            sed -i "s/6001/$PORT_1/g" /usr/local/tomcat_${VHOSTS}_${SERVER_NUM_1}/conf/
server.xml
            sed -i "s/7001/$PORT_2/g" /usr/local/tomcat_${VHOSTS}_${SERVER_NUM_1}/conf/
server.xml
            sed -i "s/8001/$PORT_3/g" /usr/local/tomcat_${VHOSTS}_${SERVER_NUM_1}/conf/
server.xml
            sed -i "/^upstream/aserver 127.0.0.1:${PORT_2} weight=1 max_fails=2 fail_
timeout=30s; " $NGINX_CONF/domains/$VHOSTS
            exit 0
```

```
        fi
        exit
    fi
        cp -r tomcat /usr/local/tomcat_ $VHOSTS
        cp -r xxx.jfedu.net $NGINX_CONF/domains/$VHOSTS
        sed -i "s/VHOSTS/$VHOSTS/g" $NGINX_CONF/domains/$VHOSTS
        sed -i "s/xxx/$VHOSTS/g" $NGINX_CONF/domains/$VHOSTS
        sed -i "s/7001/${PORT_2}/g" $NGINX_CONF/domains/$VHOSTS
    ####### config tomcat
        sed -i "s/6001/$PORT_1/g" /usr/local/tomcat_${VHOSTS}/conf/server.xml
        sed -i "s/7001/$PORT_2/g" /usr/local/tomcat_${VHOSTS}/conf/server.xml
        sed -i "s/8001/$PORT_3/g" /usr/local/tomcat_${VHOSTS}/conf/server.xml

}
if [ ! -d $NGINX_CONF -o ! -d /usr/java/$JDK_DIR ]; then
    install_nginx
    install_tomcat
fi
config_tomcat_nginx $1
```

18.10 shell 编程实战 Docker 管理脚本

Docker 虚拟化是目前主流的虚拟化解决方案，越来越多的企业在使用 Docker 轻量级虚拟化。构建、维护和管理 Docker 虚拟化平台是运维人员工作中非常重要的一个环节，开发 Docker shell 脚本可以在命令行界面快速管理和维护 Docker。

shell 脚本实现 Docker 自动安装、自动导入镜像、创建虚拟机、指定 IP 地址、将创建的 Docker 虚拟机加入 Excel 存档或者加入 MySQL 数据库，编程思路如下：

- 基于 CentOS 6.5+或者 7.X YUM 安装 Docker；
- Docker 脚本参数指定 CPU、内存、硬盘容量；
- Docker 自动检测局域网 IP 并赋予 Docker 虚拟机；
- Docker 基于 pipework 指定 IP；
- 将创建的 Docker 虚拟机信息加入至 CSV(Excel)或者 MySQL 库。

shell 脚本实现 Docker 自动安装、自动导入镜像、创建虚拟机、指定 IP 地址、将创建的 Docker 虚拟机信息加入 CSV(Excel)存档或者加入 MySQL 数据库，代码如下：

```
#!/bin/bash
#Auto install docker and Create VM
#By author jfedu.net 2017
#Define Path variables
IPADDR='ifconfig|grep -E "\<inet\>"|awk '{print $2}'|grep "192.168"|head -1'
GATEWAY='route -n|grep "UG"|awk '{print $2}'|grep "192.168"|head -1'
IPADDR_NET='ifconfig|grep -E "\<inet\>"|awk '{print $2}'|grep "192.168"|head -1|awk -F.
```

```
'{print $1"."$2"."$3"."}''
LIST="/root/docker_vmlist.csv"
if [ ! -f /usr/sbin/ifconfig ]; then
    yum install net-tools* -y
fi
for i in 'seq 1 253'; do ping -c 1 ${IPADDR_NET}${i} ; [ $? -ne 0 ]&& DOCKER_IPADDR=
"${IPADDR_NET}${i}" &&break; done >>/dev/null 2>&1
echo "##################"
echo -e "Dynamic get docker IP,The Docker IP address\n\n$DOCKER_IPADDR"
NETWORK=(
    HWADDR='ifconfig eth0|grep ether|awk '{print $2}''
    IPADDR='ifconfig eth0|grep -E "\<inet\>"|awk '{print $2}''
    NETMASK='ifconfig eth0|grep -E "\<inet\>"|awk '{print $4}''
    GATEWAY='route -n|grep "UG"|awk '{print $2}''
)
if [ -z "$1" -o -z "$2" ]; then
    echo -e "\033[32m---------------------------------\033[0m"
    echo -e "\033[32mPlease exec $0 CPU(C) MEM(G),example $0 4 8\033[0m"
    exit 0
fi
#CPU='expr $2 - 1'
if [ ! -e /usr/bin/bc ]; then
    yum install bc -y >>/dev/null 2>&1
fi
CPU_ALL='cat /proc/cpuinfo |grep processor|wc -l'
if [ ! -f $LIST ]; then
    CPU_COUNT=$1
    CPU_1="0"
    CPU1='expr $CPU_1 + 0'
    CPU2='expr $CPU1 + $CPU_COUNT - 1'
    if [ $CPU2 -gt $CPU_ALL ]; then
        echo -e "\033[32mThe System CPU count is $CPU_ALL,not more than it.\033[0m"
        exit
    fi
else
    CPU_COUNT=$1
    CPU_1='cat $LIST|tail -1|awk -F"," '{print $4}'|awk -F"-" '{print $2}''
    CPU1='expr $CPU_1 + 1'
    CPU2='expr $CPU1 + $CPU_COUNT - 1'
    if [ $CPU2 -gt $CPU_ALL ]; then
        echo -e "\033[32mThe System CPU count is $CPU_ALL,not more than it.\033[0m"
        exit
    fi
fi
MEM_F='echo $2 \* 1024|bc'
MEM='printf "%.0f\n" $MEM_F'
DISK=20
```

```
USER= $3
REMARK= $4
ping $DOCKER_IPADDR -c 1 >>/dev/null 2>&1
if [ $? -eq 0 ]; then
    echo -e "\033[32m---------------------------------\033[0m"
    echo -e "\033[32mThe IP address to be used,Please change other IP,exit.\033[0m"
    exit 0
fi
if [ ! -e /usr/bin/docker ]; then
    yum install docker* device-mapper* -y
    mkdir -p /export/docker/
    cd /var/lib/ ; rm -rf docker ; ln -s /export/docker/
    mkdir -p /var/lib/docker/devicemapper/devicemapper
    dd if=/dev/zero of=/var/lib/docker/devicemapper/devicemapper/data bs=1G count=0
seek=2000
    service docker start
    if [ $? -ne 0 ]; then
        echo "Docker install error ,please check."
        exit
    fi
fi
cd /etc/sysconfig/network-scripts/
    mkdir -p /data/backup/'date +%Y%m%d-%H%M'
    yes|cp ifcfg-eth* /data/backup/'date +%Y%m%d-%H%M'/
if
    [ -e /etc/sysconfig/network-scripts/ifcfg-br0 ]; then
    echo
else
    cat >ifcfg-eth0 <<EOF
    DEVICE=eth0
    BOOTPROTO=none
    ${NETWORK[0]}
    NM_CONTROLLED=no
    ONBOOT=yes
    TYPE=Ethernet
    BRIDGE="br0"
    ${NETWORK[1]}
    ${NETWORK[2]}
    ${NETWORK[3]}
    USERCTL=no
EOF
    cat >ifcfg-br0 <<EOF
    DEVICE="br0"
    BOOTPROTO=none
    ${NETWORK[0]}
    IPV6INIT=no
    NM_CONTROLLED=no
```

```
    ONBOOT = yes
    TYPE = "Bridge"
    ${NETWORK[1]}
    ${NETWORK[2]}
    ${NETWORK[3]}
    USERCTL = no
EOF
    /etc/init.d/network restart
fi
echo 'Your can restart Ethernet Service: /etc/init.d/network restart !'
echo '-------------------------------------------------------------'

cd -
####### create docker container
service docker status >>/dev/null
if [ $? -ne 0 ]; then
    service docker restart
fi
NAME = "Docker_'echo $DOCKER_IPADDR|awk -F"." '{print $(NF-1)"_"$NF}''"
IMAGES = 'docker images|grep -v "REPOSITORY" | grep -v "none" | grep "jfedu" | head -1 | awk '
{print $1}''
if [ -z $IMAGES ]; then
    echo "Plesae Download Docker Centos Images, you can to be use docker search centos, and
docker pull centos6.5-ssh, exit 0"
    if [ ! -f jfedu_centos68.tar ]; then
        echo "Please upload jfedu_centos68.tar for docker server."
        exit
    fi
    cat jfedu_centos68.tar|docker import - jfedu_centos6.8
fi
IMAGES = 'docker images | grep -v "REPOSITORY" | grep -v "none" | grep "jfedu" | head -1 | awk
'{print $1}''
CID = $(docker run -itd --privileged --cpuset-cpus = ${CPU1}-${CPU2} -m ${MEM}m --net =
none --name = $NAME $IMAGES /bin/bash)
echo $CID
docker ps -a |grep "$NAME"
pipework br0 $NAME $DOCKER_IPADDR/24@$IPADDR
docker exec $NAME /etc/init.d/sshd start
if [ ! -e $LIST ]; then
    echo "编号,容器 ID,容器名称,CPU,内存,硬盘,容器 IP,宿主机 IP,使用人,备注" >$LIST
fi
###################
NUM = 'cat $LIST |grep -v CPU|tail -1|awk -F, '{print $1}''
if [[ $NUM -eq "" ]]; then
        NUM = "1"
else
        NUM = 'expr $NUM + 1'
```

```
fi
####################
echo -e "\033[32mCreate virtual client Successfully.\n$NUM 'echo $CID|cut -b 1-12', $NAME,
$CPU1-$CPU2, ${MEM}M, ${DISK}G, $DOCKER_IPADDR, $IPADDR, $USER, $REMARK\033[0m"
if [ -z $USER ]; then
    USER="NULL"
    REMARK="NULL"
fi
echo $NUM,'echo $CID|cut -b 1-12', $NAME, $CPU1-$CPU2, ${MEM}M, ${DISK}G, $DOCKER_IPADDR,
$IPADDR, $USER, $REMARK >> $LIST
rm -rf /root/docker_vmlist_*
iconv -c -f utf-8 -t gb2312 $LIST -o /root/docker_vmlist_'date +%H%M'.csv
```

18.11 shell 编程实战 Bind 管理脚本

Bind 主要用于企业 DNS 平台的构建，而 DNS 用于将域名解析成 IP，用户在浏览器只需输入域名，即可访问服务器 IP 地址的虚拟主机网站。

Bind 难点在于创建各种记录，例如 A 记录、mail 记录、反向记录、资源记录，基于 shell 脚本可以减轻人工的操作，节省大量的时间成本。

shell 脚本实现 Bind 自动安装、初始化 Bind 环境、自动添加 A 记录、反向记录、批量添加 A 记录，编程思路如下：

- YUM 方式自动安装 Bind；
- 自动初始化 Bind 配置；
- 创建安装、初始化、添加记录函数；
- 自动添加单个 A 记录及批量添加 A 记录和反向记录。

shell 脚本实现 Bind 自动安装、初始化 Bind 环境、自动添加 A 记录、反向记录、批量添加 A 记录，代码如下：

```
#!/bin/bash
#Auto install config bind server
#By author jfedu.net 2017
#Define Path variables
BND_ETC=/var/named/chroot/etc
BND_VAR=/var/named/chroot/var/named
BAK_DIR=/data/backup/dns_'date +%Y%m%d-%H%M'
##Backup named server
if
      [ ! -d $BAK_DIR ]; then
      echo "Please waiting Backup Named Config ..........."
      mkdir   -p $BAK_DIR
      cp -a /var/named/chroot/{etc,var}    $BAK_DIR
      cp -a /etc/named.* $BAK_DIR
```

```
fi
##Define Shell Install Function
Install ()
{
  if
     [ ! -e /etc/init.d/named ]; then
     yum install bind* -y
else
     echo ------------------------------------------------
     echo "The Named Server is exists ,Please exit ........"
     sleep 1
fi
}
##Define Shell Init Function
Init_Config ()
{
       sed -i -e 's/localhost; /any; /g' -e '/port/s/127.0.0.1/any/g' /etc/named.conf
       echo ------------------------------------------------
       sleep 2
       echo "The named.conf config Init success !"
}
##Define Shell Add Name Function
Add_named ()
{
##DNS name
       read -p "Please Insert Into Your Add Name ,Example 51cto.com :" NAME
       echo $NAME |grep -E "com|cn|net|org"
       while
        [ "$?" -ne 0 ]
         do
        read -p "Please reInsert Into Your Add Name ,Example 51cto.com :" NAME
        echo $NAME |grep -E "com|cn|net|org"
     done
## IP address
       read -p "Please Insert Into Your Name Server IP ADDress:" IP
       echo $IP |egrep -o "([0-9]{1,3}\.){3}[0-9]{1,3}"
       while
       [ "$?" -ne "0" ]
        do
        read -p "Please reInsert Into Your Name Server IP ADDress:" IP
       echo $IP |egrep -o "([0-9]{1,3}\.){3}[0-9]{1,3}"
      done
       ARPA_IP='echo $IP|awk -F. '{print $3"."$2"."$1}''
       ARPA_IP1='echo $IP|awk -F. '{print $4}''
       cd $BND_ETC
       grep "$NAME" named.rfc1912.zones
if
```

```
        [ $? -eq 0 ]; then
        echo "The $NAME IS exist named.rfc1912.zones conf ,please exit ..."
        exit
else
        read -p "Please Insert Into SLAVE Name Server IP ADDress:" SLAVE
        echo $SLAVE |egrep -o "([0-9]{1,3}\.){3}[0-9]{1,3}"
        while
        [ "$?" -ne "0" ]
        do
            read -p "Please Insert Into SLAVE Name Server IP ADDress:" SLAVE
            echo $SLAVE |egrep -o "([0-9]{1,3}\.){3}[0-9]{1,3}"
        done
            grep "rev" named.rfc1912.zones
       if
        [ $? -ne 0 ]; then
            cat >> named.rfc1912.zones << EOF
#'date +%Y-%m-%d' Add $NAME CONFIG
zone "$NAME" IN {
        type master;
        file "$NAME.zone";
        allow-update { none; };
};
zone "$ARPA_IP.in-addr.arpa" IN {
        type master;
        file "$ARPA_IP.rev";
        allow-update { none; };
};
EOF
      else
        cat >> named.rfc1912.zones << EOF

#'date +%Y-%m-%d' Add $NAME CONFIG
zone "$NAME" IN {
        type master;
        file "$NAME.zone";
        allow-update { none; };
};
EOF
    fi
fi
       [ $? -eq 0 ]&& echo "The $NAME config name.rfc1912.zones success !"
       sleep 3 ; echo "Please waiting config $NAME zone File ............"
       cd $BND_VAR
       read -p "Please insert Name DNS A HOST ,EXample www or mail :" HOST
       read -p "Please insert Name DNS A NS IP ADDR ,EXample 192.168.111.130 :" IP_HOST
       echo $IP_HOST |egrep -o "([0-9]{1,3}\.){3}[0-9]{1,3}"
       ARPA_IP2='echo $IP_HOST|awk -F. '{print $3"."$2"."$1}''
```

```
        ARPA_IP3 = 'echo $IP_HOST|awk - F. '{print $4}''
        while
        [ "$?" - ne "0" ]
do
        read - p "Please Reinsert Name DNS A IPADDRESS ,EXample 192.168.111.130 :" IP_HOST
        echo $IP_HOST |egrep - o "([0 - 9]{1,3}\.){3}[0 - 9]{1,3}"
done
        cat > $NAME.zone << EOF
\ $TTL    86400
@               IN SOA localhost.       root.localhost. (
                                        43              ; serial (d. adams)
                                        1H              ; refresh
                                        15M             ; retry
                                        1W              ; expiry
                                        1D )            ; minimum

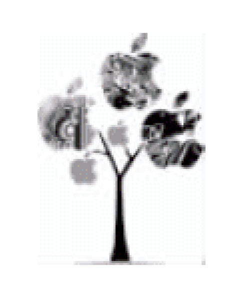
                IN NS           $NAME.
EOF
        REV = 'ls * .rev'
        ls * .rev >>/dev/null
if
        [ $? - ne 0 ]; then
        cat >> $ARPA_IP.rev << EOF
\ $TTL    86400
@       IN      SOA     localhost.      root.localhost. (
                                        1997022703      ; Serial
                                        28800           ; Refresh
                                        14400           ; Retry
                                        3600000         ; Expire
                                        86400 )         ; Minimum

          IN NS  $NAME.
EOF
         echo " $HOST              IN A           $IP_HOST" >> $NAME.zone
         echo " $ARPA_IP3          IN PTR         $HOST. $NAME. " >> $ARPA_IP.rev
         [ $? - eq 0 ]&& echo - e "The $NAME config success:\n $HOST        IN A
 $IP_HOST\n $ARPA_IP3          IN PTR          $HOST. $NAME. "
else
         sed - i "9a IN NS  $NAME."  $REV
         echo " $HOST              IN A           $IP_HOST" >           > $NAME.zone
         echo " $ARPA_IP3          IN PTR         $HOST. $NAME. " >> $REV
         [ $? - eq 0 ]&& echo - e "The $NAME config success1:\n $HOST        IN A
$IP_HOST\n $ARPA_IP3          IN PTR          $HOST. $NAME. "
fi
}
##Define Shell List A Function
Add_A_List ()
```

```
{
if
        cd $BND_VAR
        REV='ls *.rev'
        read -p "Please Insert Into Your Add Name ,Example 51cto.com :" NAME
        [ ! -e "$NAME.zone" ]; then
        echo "The $NAME.zone File is not exist ,Please ADD $NAME.zone File :"
        Add_named ;
else
        read -p "Please Enter List Name A NS File ,Example /tmp/name_list.txt: " FILE
    if
        [ -e $FILE ]; then
        for i in 'cat $FILE|awk '{print $2}'|sed "s/$NAME//g"|sed 's/\.$//g''
        #for i in 'cat $FILE|awk '{print $1}'|sed "s/$NAME//g"|sed 's/\.$//g''
do
        j='awk -v I="$i.$NAME" '{if(I==$2)print $1}' $FILE'
        echo -----------------------------------------------------------------
        echo "The $NAME.zone File is exist ,Please Enter insert NAME HOST ...."
        sleep 1
        ARPA_IP='echo $j|awk -F. '{print $3"."$2"."$1}''
        ARPA_IP2='echo $j|awk -F. '{print $4}''
        echo "$i            IN A            $j" >>$NAME.zone
        echo "$ARPA_IP2     IN PTR          $i.$NAME." >>$REV
        [ $? -eq 0 ]&& echo -e "The $NAME config success:\n$i      IN A
$j\n$ARPA_IP2         IN PTR          $i.$NAME."
done
      else
        echo "The $FILE List File IS Not Exist .......,Please exit ..."
      fi
fi
}
##Define Shell Select Menu
PS3="Please select Menu Name Config: "
select i in "自动安装 Bind 服务" "自动初始化 Bind 配置" "添加解析域名" "批量添加 A 记录"
do
case    $i    in
        "自动安装 Bind 服务")
        Install
;;
        "自动初始化 Bind 配置")
        Init_Config
;;
        "添加解析域名")
        Add_named
;;
        "批量添加 A 记录")
        Add_A_List
```

```
;;
         * )
        echo ------------------------------------------------------------
        sleep 1
        echo "Please exec: sh $0 { Install(1) or Init_Config(2) or Add_named(3) or Add_config_A(4) }"
;;
esac
done
```

第 19 章



自动化运维发展前景

随着企业服务器数量越来越多，当到达几百台、上千台之后，服务器日常管理也逐渐繁杂，每天如果通过人工频繁地更新、部署及管理这些服务器，势必会浪费大量的时间，而且有可能人为的操作也会造成某些疏忽而遗漏问题。这就需要来看一下传统的运维以及今后运维的发展方向。

本章向读者介绍如何构建企业自动化运维之路、传统运维方案及自动化运维方案、如何建立高效的 IT 自动化运维平台以及自动化运维体系各种工具等内容。

19.1 传统运维方式简介

传统的 IT 运维仍然是等到 IT 故障出现后再由运维人员采取相应的补救措施。这种被动、孤立、半自动式的 IT 运维管理模式经常让 IT 部门疲惫不堪，主要表现在以下三个方面。

(1) 运维人员被动、效率低。

在 IT 运维过程中，只有当事件已经发生并对业务已造成影响时才能发现和着手处理，这种被动“救火”不但使 IT 运维人员终日忙碌，也使 IT 运维本身质量很难提高，导致 IT 部门和业务部门对 IT 运维的服务满意度都不高。

(2) 缺乏一套高效的 IT 运维机制。

许多企业在 IT 运维管理过程中缺少自动化的运维管理模式，也没有明确的角色定义和责任划分，问题出现后很难快速、准确地找到根本原因，无法及时地找到相应的人员进行修复和处理，或者是在问题找到后缺乏流程化的故障处理机制，而在处理问题时不但欠缺规范化的解决方案，也缺乏全面的跟踪记录。

(3) 缺乏高效的 IT 运维技术工具。

随着信息化建设的深入，企业 IT 系统日趋复杂，林林总总的网络设备、服务器、中间件、业务系统等让 IT 运维人员难以从容应对，即使加班加点地维护、部署、管理也经常会因设备出现故障而导致业务的中断，严重影响企业的正常运转。

出现这些问题部分原因是企业缺乏事件监控和诊断等 IT 运维技术工具，因为在没有

高效的技术工具的支持下故障事件很难得到主动、快速地处理。

19.2 自动化运维简介

IT运维已经在风风雨雨中走过了十几个春秋，如今它正以一种全新的姿态摆在我们面前，运维自动化是IT技术发展的必然结果，现在IT系统的复杂性已经客观上要求IT运维必须能够实现数字化、自动化维护。

运维自动化是指将IT运维中日常的、大量的重复性工作自动化，把过去的手工执行转为自动化操作。自动化是IT运维工作的升华，IT运维自动化不单纯是一个维护过程，更是一个管理的提升过程，是IT运维的最高层次，也是未来的发展趋势。

19.3 运维自动化的具体内容

日常IT运维中大量的重复性工作（小到简单的日常检查、配置变更和软件安装，大到整个变更流程的组织调度）由过去的手工执行转为自动化操作，从而减少乃至消除运维中的延迟，实现"零延时"的IT运维。

简单地说，IT运维自动化是指基于流程化的框架，将事件与IT流程相关联，一旦被监控系统发现性能超标或宕机，会触发相关事件以及事先定义好的流程，可自动启动故障响应和恢复机制。

19.4 建立高效的IT自动化运维管理

建立高效的IT自动化运维管理的步骤主要包括以下几点。

（1）建立自动化运维管理平台。

IT运维自动化管理建设的第一步是要先建立IT运维的自动化监控和管理平台。通过监控工具实现对用户操作规范的约束和对IT资源进行实时监控，包括服务器、数据库、中间件、存储备份、网络、安全、机房、业务应用和客户端等内容，通过自动监控管理平台实现故障或问题综合处理和集中管理。

（2）建立故障事件自动触发流程，提高故障处理效率。

所有IT设备在遇到问题时要会自动报警，无论是系统自动报警还是使用人员报的故障，应以红色标识显示在运维屏幕上。然后IT运维人员只需要按照相关知识库的数据，一步一步操作就可以。

（3）建立规范的事件跟踪流程，强化运维执行力度。

需要建立故障和事件处理跟踪流程，利用表格工具等记录故障及其处理情况，以建立运维日志，并定期回顾从中辨识和发现问题的线索和根源。

（4）设立IT运维关键流程，引入优先处理原则。

设置自动化流程时还需要引入优先处理原则，例行的事按常规处理，特别事件要按优先级次序处理，也就是把事件细分为例行事件和例外关键事件。

19.5 IT 自动化运维工具

对于企业来说，要特别关注两类自动化工具：一是 IT 运维监控和诊断优化工具；二是运维流程自动化工具。这两类工具主要应用于如下场景：

(1) 监控自动化：是指对重要的 IT 设备实施主动式监控，如路由器、交换机、防火墙等。

(2) 配置变更检测自动化：是指 IT 设备配置参数一旦发生变化，将触发变更流程转给相关技术人员进行确认，通过自动检测协助 IT 运维人员发现和维护配置。

(3) 维护事件提醒自动化：是指通过对 IT 设备和应用活动的实时监控，当发生异常事件时系统自动启动报警和响应机制，第一时间通知相关责任人。

(4) 系统健康检测自动化：是指定期自动地对 IT 设备硬件和应用系统进行健康巡检，配合 IT 运维团队实施对系统的健康检查和监控。

(5) 维护报告生成自动化：是指定期自动地对系统做日志的收集分析，记录系统运行状况，并通过阶段性的监控、分析和总结，定时提供 IT 运维的可用性、性能、系统资源利用状况分析报告。

19.6 IT 自动化运维体系

一个完善的自动化运维体系包括系统预备、配置管理以及监控报警三个环节，每个环节实现的功能也各不相同，具体功能如下：

(1) 系统预备类。

- 自动化安装操作系统；
- 自动初始化系统；
- 自动安装各种软件包。

(2) 配置管理类。

- 自动化部署业务系统软件包并完成配置；
- 远程管理服务器；
- 配置文件、自动部署 Jenkins、网站代码变更回滚。

(3) 监控报警类。

- 服务器可用性、性能、安全监控；
- 向管理员发送报警信息。

根据提供的功能不同，自动化运维工具软件分为以下 3 类，如下 19-1 表所示。

表 19-1　自动化运维工具分类

编　号	预备类工具	配置管理类	监控报警类
1	Kickstart	Puppet	Nagios
2	Cobbler	Saltstack	Cacti
3	OpenQRM	Func	Ganglia
4	Spacewalk	Ansible	Zabbix

第 20 章



Puppet 自动运维企业实战

Puppet 是目前互联网主流三大自动化运维工具(Puppet、Ansible、Saltstack)之一，Puppet 是一种 Linux、UNIX 平台的集中配置管理系统，所谓配置管理系统，就是管理机器里面诸如文件、用户、进程、软件包等资源，其设计目标是简化对这些资源的管理以及妥善处理资源间的依赖关系。

本章向读者介绍 Puppet 工作原理、Puppet 安装配置、企业资源案例讲解、Puppet 高可用集群配置、Puppet 批量更新部署网站、Puppet+SVN 实现代码自动部署等内容。

20.1 Puppet 入门简介

Puppet 使用一种描述性语言来定义配置项，配置项中被称为“资源”，描述性语言可以声明用户的配置状态，比如声明一个软件包应该被安装或一个服务应该被启动等。

Puppet 可以运行在一台服务器端，每个客户端通过 SSL 证书连接到服务端，得到本机器的配置列表，然后根据列表来完成配置工作，所以如果硬件性能比较高，维护管理上千上万台机器是非常轻松的，前提是客户端的配置、服务器路径、软件需要保持一致。

在企业级大规模的生产环境中，如果只有一台 Puppet master，压力会非常大，因为 Puppet 是用 Ruby 语言编写的，Ruby 是解析型语言，每个客户端来访问都要解析一次，当客户端服务器很多，会造成服务器端压力很大，所以需要扩展成一个服务器集群组。

Puppet master 可以看作一个 Web 服务器，实际上也是由 Ruby 提供的 Web 服务器模块来做的。因此可以利用 Web 代理软件来配合 Puppet master 做集群设置，一般使用 Nginx+Puppet master 整合构建大型企业自动化运维管理工具，Puppet 项目主要开发者是 Luke Kanies，目前为 Puppet labs CEO，Puppet 遵循 GPLv2 版权协议。

Kanies 从 1997 年开始参与 UNIX 的系统管理工作，Puppet 的开发源于这些经验。因为对已有的配置工具不甚满意，从 2001 年到 2005 年间，Kanies 开始在 Reductive 实验室从事工具的开发。很快 Reductive 实验室发布了他们新的旗舰产品。

Puppet 是开源的基于 Ruby 的系统配置管理工具，Puppet 工作流程为 Puppet 是一个 C/S 结构，所有的 Puppet 客户端同一个服务器端的 Puppet 通信，每个 Puppet 客户端每半小时(可以设置)连接一次服务器端，下载最新的配置文件，并且严格按照配置文件来配置服

务器，配置完成以后 Puppet 客户端可以反馈给服务器端一个消息，如果报错会给服务器端反馈一个消息。

20.2 Puppet 工作原理

要熟练掌握 Puppet 在企业生产环境中的应用，需要深入理解 Puppet 服务端与客户端详细的工作流程及原理，如图 20-1 所示为 Puppet master 与 agent 完整工作流程图。

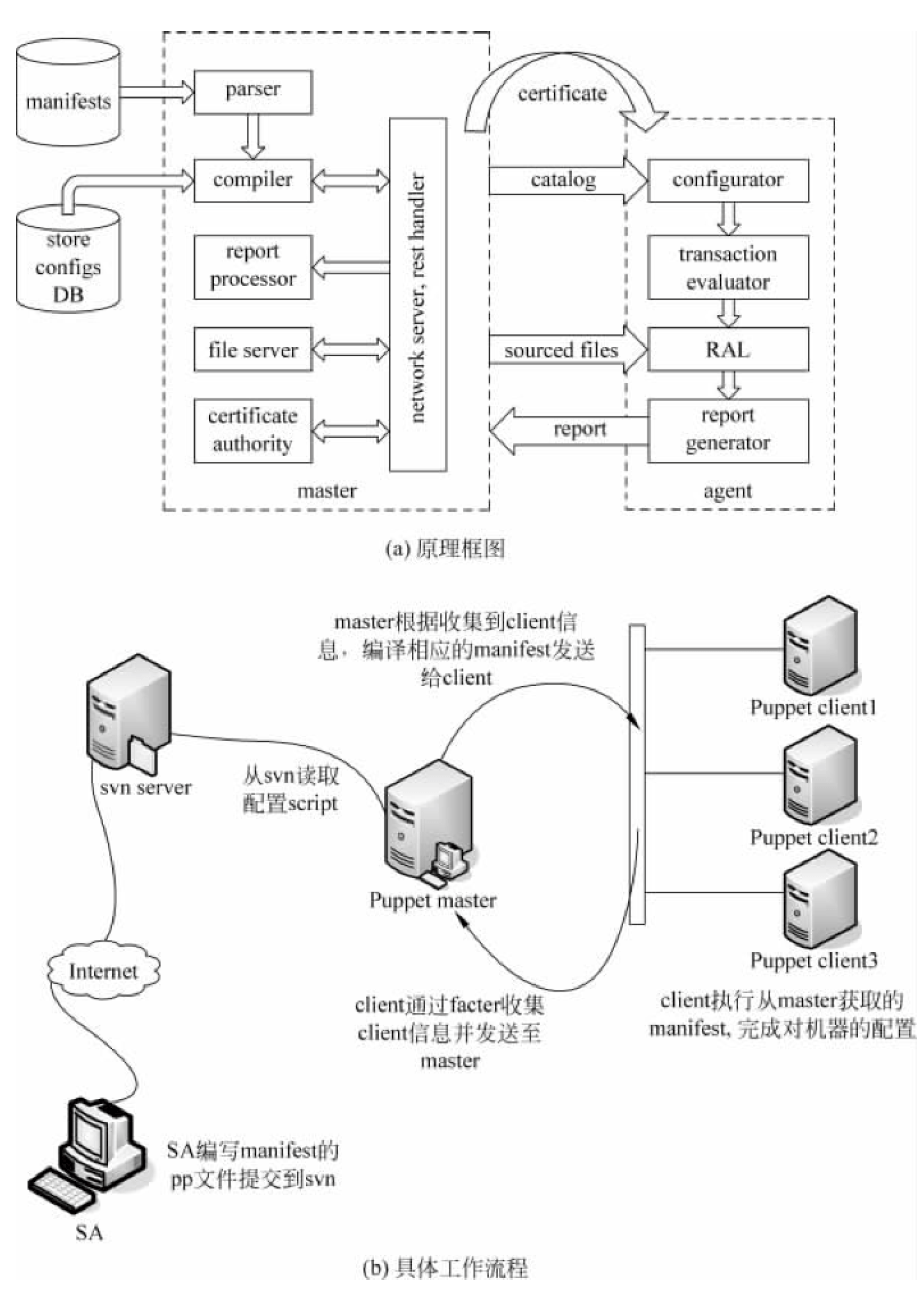


图 20-1 Puppet 工作原理图

Puppet 工作原理详解如下：

- 客户端 Puppetd 调用本地 facter，facter 会探测出该主机的常用变量，例如主机名、内存大小、IP 地址等。然后 Puppetd 把这些信息发送到 Puppet 服务端；
- Puppet 服务端检测到客户端的主机名，然后会检测 manifest 中对应的 node 配置，并对这段内容进行解析，facter 发送过来的信息可以作为变量进行处理；
- Puppet 服务端匹配 Puppet 客户端相关联的代码才能进行解析，其他的代码不解析，解析分为几个过程，首先是语法检查，然后会生成一个中间的伪代码，之后再把伪代码发给 Puppet 客户端；
- Puppet 客户端接收到伪代码之后就会执行，执行完后会将执行的结果发送给 Puppet 服务端；
- Puppet 服务端再把客户端的执行结果写入日志。

20.3　Puppet 安装配置

Puppet 工作为 C/S 模式，构建 Puppet 平台需安装 Puppet server 端和 client 端，安装之前准备好系统环境，说明如下：

- 操作系统版本：CentOS 6.5 x64；
- 服务端 ip：192.168.149.128 hostname：192-168-149-128-jfedu.net；
- 客户端 ip：192.168.149.130 hostname：192-168-149-130-jfedu.net。

（1）Puppet 服务端安装。

Puppet 服务器端需修改主机名称为 192-168-149-128-jfedu.net，并且在 hosts 文件添加主机名和本机 IP 的对应关系，如果本地局域网有 DNS 服务器，可以无须修改 hosts 文件，修改主机名及配置 hosts 代码如下：

```
hostname 'ifconfig eth0 |grep Bcast|awk '{print $2}'|cut -d: -f 2 |sed 's/\./\-/g''-jfedu.
net
cat >>/etc/hosts << EOF
192.168.149.128 192-168-149-128-jfedu.net
192.168.149.130 192-168-149-130-jfedu.net
EOF
```

Puppet 服务端除了需要安装 Puppet server 外，还需要 Ruby 的支持，需要安装 Ruby 相关软件包，默认 YUM 安装 Puppet server，会自动下载并安装 Ruby 相关软件，代码如下，详情如图 20-2 所示。

```
rpm -Uvh http://yum.puppetlabs.com/el/6/products/x86_64/puppetlabs-release-6-1.
noarch.rpm
yum install puppet-server -y
/etc/init.d/puppetmaster start
```

```
/etc/init.d/iptables stop
sed -i '/SELINUX/S/enforce/disabled/' /etc/selinux/config
setenforce 0
```

```
[root@192-168-149-128-jfedu ~]# rpm -Uvh http://yum.puppetlabs.com/
Retrieving http://yum.puppetlabs.com/el/6/products/x86_64/puppetlab
warning: /var/tmp/rpm-tmp.03j7NA: Header V4 RSA/SHA1 Signature, key
Preparing...                ########################################
   1:puppetlabs-release     ########################################
[root@192-168-149-128-jfedu ~]# yum install  puppet-server -y
Loaded plugins: fastestmirror
Determining fastest mirrors
epel/metalink
 * base: mirrors.btte.net
 * epel: mirrors.tuna.tsinghua.edu.cn
 * extras: mirrors.tuna.tsinghua.edu.cn
 * updates: mirror.bit.edu.cn
base
epel
```

图 20-2　Puppet server 服务端安装

（2）Puppet 客户端安装。

Puppet 客户端也需要修改主机名称为 192-168-149-130-jfedu.net，并且在 hosts 文件添加主机名和本机 IP 的对应关系，如果本地局域网有 DNS 服务器，可以无须修改 hosts 文件，修改主机名及配置 hosts 代码如下：

```
hostname 'ifconfig eth0 |grep Bcast|awk '{print $2}'|cut -d: -f 2 |sed 's/\./\-/g'' -jfedu.net
cat >>/etc/hosts << EOF
192.168.149.128 192-168-149-128-jfedu.net
192.168.149.130 192-168-149-130-jfedu.net
EOF
```

Puppet 客户端除了需要安装 Puppet 外，还需要 Ruby 的支持，需要安装 Ruby 相关软件包，默认 YUM 安装 Puppet，会自动下载并安装 Ruby 相关软件，代码如下，详情如图 20-3 所示。

```
rpm -Uvh http://yum.puppetlabs.com/el/6/products/x86_64/puppetlabs-release-6-1.noarch.rpm
yum install puppet -y
/etc/init.d/puppetmaster start
/etc/init.d/iptables    stop
sed -i '/SELINUX/S/enforce/disabled/' /etc/selinux/config
setenforce 0
```

（3）Puppet 客户端申请证书。

Puppet 客户端与 Puppet 服务端是通过 SSL 隧道通信的，客户端安装完成后，首次使用需向服务器端申请 Puppet 通信证书，Puppet 客户端第一次连接服务器端会发起证书申请，在 Puppet 客户端执行命令如下，返回结果如图 20-4 所示。

```
puppet agent --server 192-168-149-128-jfedu.net --test
```

```
[root@192-168-149-130-jfedu ~]# rpm -Uvh http://yum.puppetlabs.com/e

yum  install  puppet -y
/etc/init.d/puppetmaster  start
/etc/init.d/iptables      stop
sed -i  '/SELINUX/s/enforce/disabled/' /etc/selinux/config
setenforce  0
Retrieving http://yum.puppetlabs.com/el/6/products/x86_64/puppetlabs

warning: /var/tmp/rpm-tmp.3G98fV: Header V4 RSA/SHA1 Signature, key
Preparing...                ########################################
   1:puppetlabs-release     ########################################
[root@192-168-149-130-jfedu ~]# yum  install  puppet -y
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
epel/metalink
```

图 20-3　Puppet 客户端服务安装

```
-jfedu ~]#
-jfedu ~]# puppet  agent  --server 192-168-149-128-jfedu.net  --test
cate for ca
file loading from /etc/puppet/csr_attributes.yaml
SSL certificate request for 192-168-149-130-jfedu.net
uest fingerprint (SHA256): EC:89:AE:7C:42:50:5A:66:45:D1:0F:1D:4B:5F
E
cate for ca
te found and waitforcert is disabled
-jfedu ~]#
```

图 20-4　Puppet 客户端发起证书申请

（4）Puppet 服务端颁发证书。

Puppet 客户端向服务器发起证书申请，服务器端必须审核证书，如果不审核，客户端与服务器端无法进行后续正常通信，Puppet 服务端颁发证书命令代码详解如下，返回结果如图 20-5 所示。

```
[root@192-168-149-128-jfedu ~]# puppet  cert  --list
  "192-168-149-130-jfedu.net" (SHA256) EC:89:AE:7C:42:50:5A:66:45:D1:0F
7:D6:9A:10:8E
[root@192-168-149-128-jfedu ~]#
[root@192-168-149-128-jfedu ~]# puppet  cert  -s  192-168-149-130-jfedu
Notice: Signed certificate request for 192-168-149-130-jfedu.net
Notice: Removing file Puppet::SSL::CertificateRequest 192-168-149-130-j
68-149-130-jfedu.net.pem'
[root@192-168-149-128-jfedu ~]#
[root@192-168-149-128-jfedu ~]# puppet  cert  --list
[root@192-168-149-128-jfedu ~]#
[root@192-168-149-128-jfedu ~]# puppet  cert  --list --all
+ "192-168-149-128-jfedu.net" (SHA256) E5:E7:7B:2C:FE:EA:AC:04:3D:C5:21
C:23:FF:80:EE (alt names: "DNS:192-168-149-128-jfedu.net", "DNS:puppet"
+ "192-168-149-130-jfedu.net" (SHA256) C1:B5:7E:05:81:D0:BE:77:91:E3:62
0:1E:5B:DD:0C
```

图 20-5　Puppet 服务端颁发证书

- puppet cert --list：查看申请证书的客户端主机名；
- puppet cert -s 192-168-149-130-jfedu.net：颁发证书给客户端；
- puppet cert -s：为特定的主机颁发证书；
- puppet cert -s and -a：给所有的主机颁发证书；

- puppet cert --list --all：查看已经颁发的所有证书。

20.4 Puppet 企业案例演示

Puppet 是基于 C/S 架构，服务器端保存着所有对客户端服务器的配置代码，在 Puppet 服务端该配置文件叫 manifest，客户端下载 manifest 之后，可以根据 manifest 对客户端进行配置，例如软件包管理、用户管理、文件管理、命令管理、脚本管理等，Puppet 主要基于各种资源或者模块来管理客户端。

默认 Puppet 服务器端 manifest 目录在/etc/puppet/manifests/下，只需要在该目录下创建一个 site. pp 文件，然后写入相应的配置代码，Puppet 客户端跟 Puppet 服务端同步时，会检查客户端 node 配置文件，匹配之后会将该代码下载至客户端，对代码进行解析，然后在客户端执行。

以下为在 Puppet 客户端创建 test.txt 文件，并在该文件中写入测试内容，操作方法如下。

(1) Puppet 服务端创建 node 代码，创建或者编辑 vi/etc/puppet/manifests/site. pp 文件，在文件中加入如下代码：

```
node default {
file {
"/tmp/test.txt":
 content  => "Hello World,jfedu.net 2017";
   }
}
```

manifests site. pp 配置文件代码详解如下：

- node default：新建 node 节点，default 表示所有主机，可修改为特定主机名；
- file：基于 file 资源模块管理客户端文件或者目录操作；
- "/tmp/test.txt"：需在客户端文件创建的文件名；
- content：客户端服务器文件内容。

(2) 客户端执行同步命令，获取 Puppet 服务端 node 配置，代码如下，如图 20-6 所示，执行报错。

```
puppet agent --server=192-168-149-128-jfedu.net --test
```

报错原因是因为服务器端与客户端时间不同步导致的，需要同步时间，然后再次执行 puppet agent 命令，代码如下，详情如图 20-7 所示。

```
ntpdate pool.ntp.org
puppet agent --server=192-168-149-128-jfedu.net --test
```

Puppet 客户端执行同步效果，执行日志如下，会在/tmp/目录创建 test.txt 文件，内容为“Hello World,jfedu. net”，即证明 Puppet 客户端成功获取服务端 node 配置。

```
[root@192-168-149-130-jfedu tmp]# puppet  agent  --server=192-168-1
Warning: Unable to fetch my node definition, but the agent run will
Warning: SSL_connect returned=1 errno=0 state=SSLv3 read server cer
 not yet valid for /CN=Puppet CA: 192-168-149-128-jfedu.net]
Info: Retrieving pluginfacts
Error: /File[/var/lib/puppet/facts.d]: Failed to generate additiona
d=1 errno=0 state=SSLv3 read server certificate B: certificate veri
t CA: 192-168-149-128-jfedu.net]
Error: /File[/var/lib/puppet/facts.d]: Could not evaluate: Could no
jfedu.net/pluginfacts: SSL_connect returned=1 errno=0 state=SSLv3 r
certificate is not yet valid for /CN=Puppet CA: 192-168-149-128-jfe
Info: Retrieving plugin
Error: /File[/var/lib/puppet/lib]: Failed to generate additional re
errno=0 state=SSLv3 read server certificate B: certificate verify f
: 192-168-149-128-jfedu.net]
Error: /File[/var/lib/puppet/lib]: Could not evaluate: Could not re
```

图 20-6　Puppet 客户端同步服务端配置

```
[root@192-168-149-130-jfedu tmp]# puppet  agent  --server=192-168-149-
Info: Caching certificate_revocation_list for ca
Warning: Unable to fetch my node definition, but the agent run will co
Warning: undefined method `include?' for nil:NilClass
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1496805041'
Notice: /Stage[main]/Main/Node[default]/File[/tmp/test.txt]/ensure: de
38d9'
Info: Creating state file /var/lib/puppet/state/state.yaml
Notice: Finished catalog run in 0.04 seconds
[root@192-168-149-130-jfedu tmp]#
[root@192-168-149-130-jfedu tmp]# ls
test.txt
[root@192-168-149-130-jfedu tmp]# cat test.txt
Hello World, jfedu.net  2017[root@192-168-149-130-jfedu tmp]#
```

图 20-7　Puppet 客户端获取服务端 node 配置

```
Info: Caching certificate_revocation_list for ca
Warning: Unable to fetch my node definition, but the agent run will continue:
Warning: undefined method 'include?' for nil:NilClass
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1496805041'
Notice: /Stage[main]/Main/Node[default]/File[/tmp/test.txt]/ensure: defined content as '
{md5}d1c2906ad0b249a330e936e3bc1d38d9'
Info: Creating state file /var/lib/puppet/state/state.yaml
Notice: Finished catalog run in 0.04 seconds
```

20.5　Puppet 常见资源及模块

Puppet 主要基于各种资源模块管理客户端，目前企业主流 Puppet 管理客户端资源模块如下：

- file：主要负责管理文件；
- package：软件包的安装管理；

- service：系统服务的管理；
- cron：配置自动任务计划；
- exec：远程执行运行命令。

通过命令 puppet describe -l 可以查看 Puppet 支持的所有资源和模块，如图 20-8 所示。

```
[root@192-168-149-128-jfedu ~]# puppet describe -l
These are the types known to puppet:
augeas          - Apply a change or an array of changes to the  ...
computer        - Computer object management using DirectorySer ...
cron            - Installs and manages cron jobs
exec            - Executes external commands
file            - Manages files, including their content, owner ...
filebucket      - A repository for storing and retrieving file  ...
group           - Manage groups
host            - Installs and manages host entries
interface       - This represents a router or switch interface
k5login         - Manage the `.k5login` file for a user
macauthorization - Manage the Mac OS X authorization database
mailalias       - .. no documentation ..
maillist        - Manage email lists
```

(a) Puppet支持的资源及模块(1)

```
router          - .. no documentation ..
schedule        - Define schedules for Puppet
scheduled_task  - Installs and manages Windows Scheduled Tasks
selboolean      - Manages SELinux booleans on systems with SELi ...
selmodule       - Manages loading and unloading of SELinux poli ...
service         - Manage running services
ssh_authorized_key - Manages SSH authorized keys
sshkey          - Installs and manages ssh host keys
stage           - A resource type for creating new run stages
tidy            - Remove unwanted files based on specific crite ...
user            - Manage users
vlan            - .. no documentation ..
whit            - Whits are internal artifacts of Puppet's curr ...
yumrepo         - The client-side description of a yum reposito ...
zfs             - Manage zfs
zone            - Manages Solaris zones
```

(b) Puppet支持的资源及模块(2)

图 20-8 Puppet 支持的资源及模块

通过命令 puppet describe -s file 可以查看 Puppet file 资源所有的帮助信息，如图 20-9 所示。

```
[root@192-168-149-128-jfedu ~]# puppet describe -s file

file
====
Manages files, including their content, ownership, and permissions.
The `file` type can manage normal files, directories, and symlinks;
type should be specified in the `ensure` attribute.
File contents can be managed directly with the `content` attribute,
downloaded from a remote source using the `source` attribute; the la
can also be used to recursively serve directories (when the `recurse
attribute is set to `true` or `local`). On Windows, note that file
contents are managed in binary mode; Puppet never automatically tran
line endings.
**Autorequires:** If Puppet is managing the user or group that owns
file, the file resource will autorequire them. If Puppet is managing
parent directories of a file, the file resource will autorequire the
```

(a) Puppet file资源模块详情(1)

图 20-9 Puppet file 资源模块详情

```
**Autorequires:** If Puppet is managing the user or group that owns a
file, the file resource will autorequire them. If Puppet is managing any
parent directories of a file, the file resource will autorequire them.

Parameters
----------
    backup, checksum, content, ctime, ensure, force, group, ignore, links
    mode, mtime, owner, path, purge, recurse, recurselimit, replace,
    selinux_ignore_defaults, selrange, selrole, seltype, seluser, show_di
    source, source_permissions, sourceselect, target, type, validate_cmd,
    validate_replacement

Providers
---------
    posix, windows
```

(b) Puppet file资源模块详情(2)

图 20-9　（续）

20.6　Puppet file 资源案例

Puppet file 资源主要用于管理客户端文件，包括文件的内容、所有权和权限，其可管理的文件类型包括普通文件、目录以及符号链接等。

类型应在“确保”属性中指定，如果是文件内容可以直接用 content 属性来管理，或者使用 source 属性从远程源下载，后者也可以用 recurse 服务目录（当 recurse 属性设置为 true 或 local 时。Puppet file 资源支持参数详解如下：

- ensure：默认为文件或目录。
- backup：通过 filebucket 备份文件。
- checksum：检查文件是否被修改的方法。
- ctime：只读属性，文件的更新时间。
- mtime：只读属性，文件的修改时间。
- content：文件的内容，与 source 和 target 互斥。
- force：强制执行删除文件、软链接及目录的操作。
- owner：用户名或用户 ID。
- group：指定文件名的用户组或组 ID。
- link：软链接。
- mode：文件权限配置，通常采用数字符号。
- path：文件路径。
- parameters：backup，checksum，content，ctime，ensure，force，group，ignore，links，mode，mtime，owner，path，purge，recurse，recurselimit，replace，selinux_ignore_defaults，selrange，selrole，seltype，seluser，show_diff，source，source_permissions，sourceselect，target，type，validate_cmd，validate_replacement。
- providers：posix，windows。

（1）从 Puppet 服务器下载 nginx. conf 文件至客户端/tmp 目录，首先需要将 nginx. conf 文件 cp 至/etc/puppet/files 目录，然后在/etc/puppet/fileserver. conf 中添加以下三行代码，并重启 Puppet master 即可。

```
[files]
path /etc/puppet/files/
allow *
```

创建 site. pp 文件，代码如下：

```
node default {
    file {
            '/tmp/nginx.conf':
            mode => '644',
            owner => 'root',
            group => 'root',
            source => 'puppet://192-168-149-128-jfedu.net/files/nginx.conf',
    }
}
```

客户端同步配置，如图 20-10 所示。

```
[root@192-168-149-130-jfedu tmp]#
[root@192-168-149-130-jfedu tmp]# puppet  agent  --server=192-168-
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1496972457'
Notice: /Stage[main]/Main/Node[default]/File[/tmp/nginx.conf]/ensu
md5}349368d93dc27680f546b5a4991aa2ff'
Notice: Finished catalog run in 0.22 seconds
[root@192-168-149-130-jfedu tmp]# ll
total 8
-rw-r--r-- 1 root root 2650 Jun  9 09:40 nginx.conf
-rw-r--r-- 1 root root   29 Jun  9 08:58 test.txt
[root@192-168-149-130-jfedu tmp]#
```

图 20-10 Puppet file 资源远程下载文件

（2）从 Puppet 服务器下载 sysctl. conf，如果客户端该文件存在则备份为 sysctl. conf. bak，然后再覆盖原文件，site. pp 代码如下，详情如图 20-11 所示。

```
node default {
    file {
            "/etc/sysctl.conf":
            source => "puppet://192-168-149-128-jfedu.net/files/sysctl.conf",
            backup => ".bak_$uptime_seconds",
    }
}
```

（3）在 agent 上创建/export/docker 的软链接为/var/lib/docker/，site. pp 代码如下，详情如图 20-12 所示。

```
[root@192-168-149-130-jfedu ~]# puppet  agent  --server=192-168-149-
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1496973190'
Notice: /Stage[main]/Main/Node[default]/File[/etc/sysctl.conf]/conte
--- /etc/sysctl.conf    2017-05-26 20:48:35.582406831 +0800
+++ /tmp/puppet-file20170609-4129-1mi9ifq-0      2017-06-09 09:53:10.
@@ -1,30 +1,33 @@
 net.ipv4.ip_forward = 0
-
-# Controls source route verification
 net.ipv4.conf.default.rp_filter = 1
-
-# Do not accept source routing
 net.ipv4.conf.default.accept_source_route = 0
```

图 20-11　Puppet file 资源备份文件

```
node default {
    file {
            "/var/lib/docker":
            ensure  => link,
            target  => "/export/docker",
    }
}
```

```
@192-168-149-130-jfedu ~]#
@192-168-149-130-jfedu ~]# puppet  agent  --server=192-168-149-128-jfedu.
 Retrieving pluginfacts
 Retrieving plugin
 Caching catalog for 192-168-149-130-jfedu.net
 Applying configuration version '1496974040'
e: /Stage[main]/Main/Node[default]/File[/var/lib/docker]/ensure: created
e: Finished catalog run in 0.16 seconds
@192-168-149-130-jfedu ~]#
@192-168-149-130-jfedu ~]# ll /var/lib/|grep docker
wxrwx   1 root    root      14 Jun  9 10:07 docker -> /export/docker
@192-168-149-130-jfedu ~]#
@192-168-149-130-jfedu ~]#
```

图 20-12　Puppet file 资源备份文件

（4）在 agent 上创建目录/tmp/20501212，site. pp 代码如下，详情如图 20-13 所示。

```
node default {
    file {
            "/tmp/20501212":
            ensure  => directory;
    }
}
```

```
192.168.149.130  ×
168-149-130-jfedu ~]#
168-149-130-jfedu ~]# puppet  agent  --server=192-168-149-128-j
ieving pluginfacts
ieving plugin
ing catalog for 192-168-149-130-jfedu.net
ying configuration version '1496974169'
tage[main]/Main/Node[default]/File[/tmp/20501212]/ensure: creat
nished catalog run in 0.14 seconds
168-149-130-jfedu ~]#
```

图 20-13　Puppet file 创建目录

20.7 Puppet package 资源案例

Puppet package 资源主要用于管理客户端服务器的软件包，YUM 源为/etc/yum.repo.d/安装和升级操作，通过 Puppet 基于 YUM 自动安装软件包，所以需要先配置好 YUM 源。

常见的操作可以对软件包进行安装、卸载以及升级操作。Puppet package 资源支持参数详解如下：

- parameters：adminfile，allow_virtual，allowcdrom，category，configfiles，description，ensure，flavor，install_options，instance，name，package_settings，platform，responsefile，root，source，status，uninstall_options，vendor。
- providers：aix，appdmg，apple，apt，aptitude，aptrpm，blastwave，dpkg，fink，freebsd，gem，hpux，macports，msi，nim，openbsd，opkg，pacman，pip，pkg，pkgdmg，pkgin，pkgutil，portage，ports，portupgrade，rpm，rug，sun，sunfreeware，up2date，urpmi，windows，yum，zipper。
- present：检查软件是否存在，不存在则安装。
- installed：表示安装软件。
- absent：删除(无依赖)，当别的软件包依赖时，不可删除。
- pureged：删除所有配置文件和依赖包，有潜在风险，慎用。
- latest：升级到最新版本。
- version：指定安装具体的某个版本号。

(1) 客户端安装 ntpdate 及 screen 软件，代码如下，执行结果如图 20-14 所示。

```
node default {
    package {
            ["screen","ntp"]:
            ensure => "installed";
    }
}
```

```
[root@192-168-149-130-jfedu ~]# ntpdate
-bash: /usr/sbin/ntpdate: No such file or directory
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]# puppet  agent  --server=192-168-149-128-
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1496975410'
Notice: /Stage[main]/Main/Node[default]/Package[screen]/ensure: created
Notice: /Stage[main]/Main/Node[default]/Package[ntpdate]/ensure: created
Notice: Finished catalog run in 4.32 seconds
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]# ntpdate
 9 Jun 10:30:55 ntpdate[6044]: no servers can be used, exiting
```

图 20-14 Puppet package 安装软件

（2）客户端卸载 ntpdate 及 screen 软件，代码如下，执行结果如图 20-15 所示。

```
node default {
    package {
            ["screen","ntp"]:
            ensure => "absent";
    }
}
```

```
192.168.149.130 ×
168-149-130-jfedu ~]#
168-149-130-jfedu ~]# puppet  agent  --server=192-168-149-128-jfedu.r
ieving pluginfacts
ieving plugin
ing catalog for 192-168-149-130-jfedu.net
ying configuration version '1496975573'
kage[main]/Main/Node[default]/Package[screen]/ensure: removed
kage[main]/Main/Node[default]/Package[ntpdate]/ensure: removed
nished catalog run in 0.63 seconds
168-149-130-jfedu ~]#
```

图 20-15　Puppet package 卸载软件

20.8　Puppet service 资源案例

Puppet service 资源主要用于启动、重启和关闭客户端的守护进程，同时可以监控进程的状态，还可以将守护进程加入到自启动中。Puppet service 资源支持参数详解如下：

- parameters：binary，control，enable，ensure，flags，hasrestart，hasstatus，manifest，name，path，pattern，restart，start，status，stop。
- providers：base，bsd，daemontools，debian，freebsd，gentoo，init，launchd，openbsd，openrc，openwrt，redhat，runit，service，smf，src，systemd，upstart，windows。
- enable：指定服务在开机的时候是否启动，可以设置 true 和 false。
- ensure：是否运行服务，running 表示运行，stopped 表示停止服务。
- name：守护进程的名字。
- path：启动脚本搜索路径。
- provider：默认为 init。
- hasrestart：管理脚本是否支持 restart 参数，如果不支持，就用 stop 和 start 实现 restart 效果。
- hasstatus：管理脚本是否支持 status 参数，Puppet 用 status 参数来判断服务是否已经在运行了，如果不支持 status 参数，Puppet 利用查找运行进程列表里面是否有服务名来判断服务是否在运行。

（1）启动 agent httpd 服务，停止 nfs 服务，代码如下，结果如图 20-16 所示。

```
node default {
    service {
            "httpd":
            ensure => running;
            "nfs":
            ensure => stopped;
    }
}
```

```
root      6399     2  0 11:10 ?        00:00:00 [nfsd]
root      6400     2  0 11:10 ?        00:00:00 [nfsd]
root      6459  1061  0 11:10 pts/0    00:00:00 grep -E httpd|nfs
[root@192-168-149-130-jfedu ~]# puppet  agent  --server=192-168-149-128-jfe
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1496977872'
Notice: /Stage[main]/Main/Node[default]/Service[nfs]/ensure: ensure changed
topped'
Notice: /Stage[main]/Main/Node[default]/Service[httpd]/ensure: ensure chang
'running'
Info: /Stage[main]/Main/Node[default]/Service[httpd]: Unscheduling refresh
]
Notice: Finished catalog run in 1.37 seconds
```

(a) Puppet service重启服务(1)

```
[root@192-168-149-130-jfedu ~]# ps -ef |grep http
root      6725     1  0 11:11 ?        00:00:00 /usr/sbin/httpd
apache    6737  6725  0 11:11 ?        00:00:00 /usr/sbin/httpd
apache    6738  6725  0 11:11 ?        00:00:00 /usr/sbin/httpd
apache    6739  6725  0 11:11 ?        00:00:00 /usr/sbin/httpd
apache    6740  6725  0 11:11 ?        00:00:00 /usr/sbin/httpd
apache    6741  6725  0 11:11 ?        00:00:00 /usr/sbin/httpd
apache    6742  6725  0 11:11 ?        00:00:00 /usr/sbin/httpd
apache    6743  6725  0 11:11 ?        00:00:00 /usr/sbin/httpd
apache    6744  6725  0 11:11 ?        00:00:00 /usr/sbin/httpd
root      6756  1061  0 11:13 pts/0    00:00:00 grep http
[root@192-168-149-130-jfedu ~]# ps -ef |grep nfs
root      6761  1061  0 11:13 pts/0    00:00:00 grep nfs
[root@192-168-149-130-jfedu ~]#
```

(b) Puppet service重启服务(2)

图 20-16　Puppet service 重启服务

（2）启动 agent httpd 服务并且开机启动，停止 nfs 服务，开机不启动，代码如下，结果如图 20-17 所示。

```
node default {
    service {
            "httpd":
            ensure => running,
            enable => true;
            "nfs":
            ensure => stopped,
            enable => false;
    }
}
```

```
[root@192-168-149-130-jfedu ~]# chkconfig --list nfs
nfs             0:off   1:off   2:on    3:on    4:on    5:on    6:off
[root@192-168-149-130-jfedu ~]# chkconfig --list httpd
httpd           0:off   1:off   2:off   3:off   4:off   5:off   6:off
[root@192-168-149-130-jfedu ~]# puppet  agent  --server=192-168-149-12
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1496978179'
Notice: /Stage[main]/Main/Node[default]/Service[nfs]/enable: enable ch
e'
Notice: /Stage[main]/Main/Node[default]/Service[httpd]/enable: enable
rue'
Notice: Finished catalog run in 0.55 seconds
[root@192-168-149-130-jfedu ~]#
```

(a) Puppet service开机启动(1)

```
Notice: /Stage[main]/Main/Node[default]/Service[nfs]/enable: enable ch
e'
Notice: /Stage[main]/Main/Node[default]/Service[httpd]/enable: enable
rue'
Notice: Finished catalog run in 0.55 seconds
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]# chkconfig --list httpd
httpd           0:off   1:off   2:on    3:on    4:on    5:on    6:off
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]# chkconfig --list nfs
nfs             0:off   1:off   2:off   3:off   4:off   5:off   6:off
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]#
```

(b) Puppet service开机启动(2)

图 20-17　Puppet service 开机启动

20.9　Puppet exec 资源案例

Puppet exec 资源主要用于客户端远程执行命令或者软件安装等，相当于 shell 的调用，exec 是一次性执行资源，在不同类里面 exec 名字可以相同。Puppet exec 资源支持参数如下：

- parameters：command，creates，cwd，environment，group，logoutput，onlyif，path，refresh，refreshonly，returns，timeout，tries，try_sleep，umask，unless，user。
- providers：posix，shell，windows。
- command：指定要执行的系统命令。
- creates：指定命令所生成的文件。
- cwd：指定命令执行目录，如果目录不存在，则命令执行失败。
- group：执行命令运行的账户组。
- logoutput：是否记录输出。
- onlyif：exec 只会在 onlyif 设定的命令返回 0 时才执行。
- path：命令执行的搜索路径。
- refresh => true|false：刷新命令执行状态。
- refreshonly => true|false：该属性可以使命令变成仅刷新触发的。

- returns：指定返回的代码。
- timeout：命令运行的最长时间。
- tries：命令执行重试次数，默认为 1。
- try_sleep：设置命令重试的间隔时间，单位为 s。
- user：指定执行命令的账户。
- provider：shell 和 windows。
- environment：为命令设定额外的环境变量，要注意的是如果设定 path，path 的属性会被覆盖。

(1) agent 服务器执行 tar 解压 Nginx 软件包，代码如下，结果如图 20-18 所示。

```
node default {
    exec {
            'Agent tar xzf nginx-1.12.0.tar.gz':
            path => ["/usr/bin","/bin"],
            user => 'root',
            group => 'root',
            timeout => '10',
            command => 'tar -xzf /tmp/nginx-1.12.0.tar.gz',
    }
}
```

```
[root@192-168-149-130-jfedu tmp]# ll /var/log/nginx.log
ls: cannot access /var/log/nginx.log: No such file or directory
[root@192-168-149-130-jfedu tmp]#
[root@192-168-149-130-jfedu tmp]# puppet  agent  --server=192-168-149-128-jf
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1496979312'
Notice: /Stage[main]/Main/Node[default]/Exec[Agent tar xzf nginx-1.6.2.tar.g
cuted successfully
Notice: Finished catalog run in 0.65 seconds
[root@192-168-149-130-jfedu tmp]# ls
20170808  20501212  nginx-1.6.2  nginx-1.6.2.tar.gz  nginx.conf  test.txt
[root@192-168-149-130-jfedu tmp]#
```

图 20-18　Puppet exec 远程执行命令

(2) agent 服务器远程执行 auto_install_nginx.sh 脚本，代码如下，结果如图 20-19 所示。

```
node default {
    file {
            "/tmp/auto_install_nginx.sh":
            source =>"puppet://192-168-149-128-jfedu.net/files/auto_install_nginx.sh",
            owner => "root",
            group => "root",
            mode => 755,
    }
    exec {
            "/tmp/auto_install_nginx.sh":
            cwd => "/tmp",
```

```
            user => root,
            path => ["/usr/bin","/usr/sbin","/bin","/bin/sh"],
        }
    }
```

```
[root@192-168-149-130-jfedu tmp]#
[root@192-168-149-130-jfedu tmp]# ls /usr/local/nginx/
ls: cannot access /usr/local/nginx/: No such file or directory
[root@192-168-149-130-jfedu tmp]#
[root@192-168-149-130-jfedu tmp]# puppet  agent  --server=192-168-1
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1496980772'
Notice: /Stage[main]/Main/Node[default]/Exec[/tmp/auto_install_ngin
Notice: Finished catalog run in 54.97 seconds
[root@192-168-149-130-jfedu tmp]# ls /usr/local/nginx/
conf  html  logs  sbin
[root@192-168-149-130-jfedu tmp]#
```

图 20-19　Puppet exec 执行 Nginx 安装脚本

（3）agent 服务器更新 sysctl. conf，如果该文件发生改变，则执行命令 sysctl -p，代码如下，结果如图 20-20 所示。

```
node default {
    file {
            "/etc/sysctl.conf":
            source =>"puppet://192-168-149-128-jfedu.net/files/sysctl.conf",
            owner => "root",
            group => "root",
            mode => 644,
    }
    exec {
            "sysctl refresh kernel config":
            path => ["/usr/bin", "/usr/sbin", "/bin", "/sbin"],
            command => "/sbin/sysctl -p",
            subscribe => File["/etc/sysctl.conf"],
            refreshonly => true
    }
}
```

```
[root@192-168-149-130-jfedu ~]# puppet  agent  --server=192-168-149-128
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1496979816'
Notice: /Stage[main]/Main/Node[default]/Exec[Agent tar xzf nginx-1.6.2.
Notice: /Stage[main]/Main/Node[default]/File[/etc/sysctl.conf]/content:
--- /etc/sysctl.conf    2017-06-09 11:44:13.357579239 +0800
+++ /tmp/puppet-file20170609-7827-kmfu63-0      2017-06-09 11:44:19.732
@@ -1,3 +1,4 @@
+net.ipv4.ip_forward = 0
 net.ipv4.conf.default.rp_filter = 1
 net.ipv4.conf.default.accept_source_route = 0
 kernel.sysrq = 0
```

(a) Puppet exec更新执行触发命令(1)

图 20-20　Puppet exec 更新执行触发命令

```
Info: Computing checksum on file /etc/sysctl.conf
Info: FileBucket got a duplicate file {md5}f6839b49be2828e7416c50f3e5aa
Info: /Stage[main]/Main/Node[default]/File[/etc/sysctl.conf]: Filebucke
2828e7416c50f3e5aa67c6
Notice: /Stage[main]/Main/Node[default]/File[/etc/sysctl.conf]/content:
a67c6' to '{md5}2a159a404fb0e33557d83bae50c467f0'
Info: /Stage[main]/Main/Node[default]/File[/etc/sysctl.conf]: Schedulin
Notice: /Stage[main]/Main/Node[default]/Exec[sysctl refresh kernel conf
Notice: Finished catalog run in 0.87 seconds
[root@192-168-149-130-jfedu ~]#
```

(b) Puppet exec更新执行触发命令(2)

图 20-20 （续）

20.10 Puppet cron 资源案例

Puppet cron 资源主要用于安装和管理 crontab 计划任务，每一个 cron 资源需要一个 command 属性和 user 属性以及至少一个周期属性（hour、minute、month、monthday、weekday）。

crontab 计划任务的名称不是计划任务的一部分，它是 Puppet 用来存储和检索该资源。假如用户指定了一个除了名称其他的都和一个已经存在的计划任务相同，那么这两个计划任务被认为是等效的，并且新名称将会永久地与该计划任务相关联。Puppet cron 资源支持参数详解如下：

- parameters：command，ensure，environment，hour，minute，month，monthday，name，special，target，user，weekday。
- providers：crontab。
- user：加某个用户的 crontab 任务，默认是运行 Puppet 的用户。
- command：要执行的命令或脚本路径，可不写，默认是 title 名称。
- ensure：表示该资源是否启用，可设置成 true 或 false。
- environment：crontab 环境里面指定环境变量。
- hour：设置 crontab 的小时，可设置成 0～23。
- minute：指定 crontab 的分钟，可设置成 0～59。
- month：指定 crontab 运行的月份，可设置成 1～12。
- monthday：指定月的天数可设置成 1～31。
- name：crontab 的名字，区分不同的 crontab。
- provider：可用的 provider 有 crontab 默认的 crontab 程序。
- target：crontab 作业存放的位置。
- weekday：设置 crontab 的星期数，可设置成 0～7，其中周日为 0。

（1）agent 服务器添加 ntpdate 时间同步任务，代码如下，结果如图 20-21 所示。

```
node default {
```

```
    cron{
            "ntpdate":
            command => "/usr/sbin/ntpdate pool.ntp.org",
            user => root,
            hour => 0,
            minute => 0,
    }
}
```

```
[root@192-168-149-130-jfedu ~]# puppet  agent  --server=192-168-149-1
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1496985201'
Notice: /Stage[main]/Main/Node[default]/Cron[ntpdate]/ensure: created
Notice: Finished catalog run in 0.46 seconds
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]# crontab  -l
# HEADER: This file was autogenerated at Fri Jun 09 13:13:22 +0800 20
# HEADER: While it can still be managed manually, it is definitely no
# HEADER: Note particularly that the comments starting with 'Puppet N
# HEADER: not be deleted, as doing so could cause duplicate cron jobs
# Puppet Name: ntpdate
0 0 * * * /usr/sbin/ntpdate pool.ntp.org
[root@192-168-149-130-jfedu ~]#
```

图 20-21　Puppet cron 创建任务计划

(2) agent 服务器删除 ntpdate 时间同步任务，代码如下，结果如图 20-22 所示。

```
node default {
    cron{
            "ntpdate":
            command => "/usr/sbin/ntpdate pool.ntp.org",
            user => root,
            hour => 0,
            minute => 0,
            ensure => absent,
    }
}
```

```
# HEADER: not be deleted, as doing so could cause duplicate cron jobs.
# Puppet Name: ntpdate
0 0 * * * /usr/sbin/ntpdate pool.ntp.org
[root@192-168-149-130-jfedu ~]# puppet  agent  --server=192-168-149-128-j
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1496988132'
Notice: /Stage[main]/Main/Node[default]/Cron[ntpdate]/ensure: removed
Notice: Finished catalog run in 0.53 seconds
[root@192-168-149-130-jfedu ~]# crontab  -l
# HEADER: This file was autogenerated at Fri Jun 09 14:02:13 +0800 2017 b
# HEADER: While it can still be managed manually, it is definitely not re
# HEADER: Note particularly that the comments starting with 'Puppet Name'
# HEADER: not be deleted, as doing so could cause duplicate cron jobs.

[root@192-168-149-130-jfedu ~]#
```

图 20-22　Puppet cron 删除任务计划

20.11 Puppet 日常管理与配置

Puppet 平台构建完毕，能够使用 Puppet 去管理客户端，对文件、服务、脚本、各种配置的变更，如果要管理批量服务器，还需要进行一些步骤的配置。

20.11.1 Puppet 自动认证

企业新服务器通过 kickstart 自动安装 Linux 操作系统，安装完毕，可以自动安装 Puppet 相关软件包，Puppet 客户端安装完毕，需向 Puppet 服务端请求证书，然后 Puppet 服务端颁发证书给客户端，默认需要手动颁发，可以通过配置让 Puppet 服务端自动颁发证书。

自动颁发证书前提是服务端与客户端能 ping 通彼此的主机名，配置自动颁发证书需在 Puppet 服务器端的 puppet.conf 配置文件 main 段加入如下代码，详情如图 20-23 所示。

```
[main]
autosign = true
```

```
[root@192-168-149-128-jfedu ~]# vim /etc/puppet/puppet.conf
########
[main]
    # The Puppet log directory.
    # The default value is '$vardir/log'.
    logdir = /var/log/puppet

    # Where Puppet PID files are kept.
    # The default value is '$vardir/run'.
    rundir = /var/run/puppet

    autosign = true

    # Where SSL certificates are kept.
    # The default value is '$confdir/ssl'.
```

图 20-23 Puppet 服务端添加自动颁发证书

重启 Puppet master 服务，并且删除 192.168.149.130 证书，代码如下：

```
/etc/init.d/puppetmaster restart
puppet cert --clean 192-168-149-130-jfedu.net
```

删除 Puppet 客户端 SSL 文件，重新生成 SSL 文件，执行如下命令自动申请证书：

```
rm -rf /var/lib/puppet/ssl/
puppet agent --server=192-168-149-128-jfedu.net --test
```

Puppet 服务端会自动认证，即服务器端不必手动颁发证书，减轻人工的干预和操作，如图 20-24 所示。

20.11.2 Puppet 客户端自动同步

Puppet 客户端安装完，并且认证完之后，如果在 Puppet 服务端配置了 node 信息，客户

```
[root@192-168-149-130-jfedu ~]# puppet  agent  --server=192-168-149-128-jf
Info: Creating a new SSL key for 192-168-149-130-jfedu.net
Info: Caching certificate for ca
Info: csr_attributes file loading from /etc/puppet/csr_attributes.yaml
Info: Creating a new SSL certificate request for 192-168-149-130-jfedu.net
Info: Certificate Request fingerprint (SHA256): 65:F6:C0:E9:26:DF:91:5C:33
2:D5:FD:19:10:D9:EC:25
Info: Caching certificate for 192-168-149-130-jfedu.net
Info: Caching certificate_revocation_list for ca
Info: Caching certificate for ca
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Caching catalog for 192-168-149-130-jfedu.net
Info: Applying configuration version '1497064668'
Notice: Finished catalog run in 0.08 seconds
[root@192-168-149-130-jfedu ~]#
```

图 20-24 Puppet 客户端自动获取证书

端启动服务，默认 30min 自动与服务端同步信息，如果要修改同步的时间频率，修改 Puppet 客户端配置信息即可。

Puppet 客户端配置 Puppet 相关参数和同步时间，修改/etc/sysconfig/puppet 配置文件，最终代码如下：

```
# The puppetmaster server
PUPPET_SERVER = 192 - 168 - 149 - 128 - jfedu.net
# If you wish to specify the port to connect to do so here
PUPPET_PORT = 8140
# Where to log to. Specify syslog to send log messages to the system log.
PUPPET_LOG = /var/log/puppet/puppet.log
# You may specify other parameters to the puppet client here
PUPPET_EXTRA_OPTS = -- waitforcert = 500
```

/etc/sysconfig/puppet 配置文件参数详解如下。

- PUPPET_SERVER=192-168-149-128-jfedu.net：指定 Puppet master 主机名。
- PUPPET_PORT=8140：指定 Puppet master 端口。
- PUPPET_LOG=/var/log/puppet/puppet.log：Puppet 客户端日志路径。
- PUPPET_EXTRA_OPTS=--waitforcert=500：获取 Puppet master 证书返回等待时间。

重启 Puppet 客户端服务，客户端会半个小时跟服务器同步一次配置信息，代码如下：

```
/etc/init.d/puppet restart
```

可以修改与服务器端同步配置信息的时间，修改 vi /etc/puppet/puppet.conf 文件，在[agent]段加入如下语句，表示 60s 与 Puppet master 同步一次配置信息，重启 Puppet，同步结果如图 20-25 所示。

```
[agent]
runinterval = 60
```

20.11.3 Puppet 服务端主动推送

上述 Puppet 客户端配置每 60s 与服务端同步配置信息，如果服务器端更新了配置信

```
[root@192-168-149-130-jfedu tmp]# /etc/init.d/puppet restart
Stopping puppet agent:                                          [FA
Starting puppet agent:                                          [
[root@192-168-149-130-jfedu tmp]#
[root@192-168-149-130-jfedu tmp]# tail -fn 100 /var/log/puppet
Sat Jun 10 11:36:01 +0800 2017 Puppet (notice): Reopening log
Sat Jun 10 11:36:01 +0800 2017 Puppet (notice): Starting Puppe
Sat Jun 10 11:36:02 +0800 2017 Puppet (notice): Finished catal
-------------------
Sat Jun 10 11:37:02 +0800 2017 /Stage[main]/Main/Node[default]
'{md5}349368d93dc27680f546b5a4991aa2ff'
Sat Jun 10 11:37:02 +0800 2017 Puppet (notice): Finished catal
-------------------
Sat Jun 10 11:38:02 +0800 2017 /Stage[main]/Main/Node[default]
ed '0755' to '0644'
Sat Jun 10 11:38:02 +0800 2017 Puppet (notice): Finished catal
```

图 20-25　Puppet 客户端自动同步服务端配置

息，想立刻让客户端同步，如何通知客户端来获取最新的配置信息呢？可以使用 Puppet master 主动推送的方式，让客户端更新服务端最新的配置信息。

Puppet 服务器端使用 puppet run 命令可以给客户端发送一段信号，告诉客户端立刻跟服务器同步配置信息，配置方法如下。

（1）修改 Puppet 客户端配置文件/etc/puppet/puppet.conf，在 agent 段加入如下代码：

```
[agent]
listen = true
```

（2）修改 Puppet 客户端配置文件/etc/sysconfig/puppet，指定 Puppet master 端主机名，代码如下：

```
PUPPET_SERVER = 192 - 168 - 149 - 128 - jfedu.net
```

（3）创建 Puppet 客户端配置文件 namespaceauth.conf，写入如下代码：

```
[puppetrunner]
allow *
```

（4）修改 Puppet 客户端配置文件 auth.conf，在 path /前添加如下代码：

```
path /run
method save
allow *
```

（5）重启 Puppet 客户端，代码如下：

```
/etc/init.d/puppet restart
```

（6）Puppet 服务端执行如下命令，通知客户端来同步配置，也可以批量通知其他客户端，只需将客户端的主机名写入 host.txt 文件，代码如下，详情如图 20-26 所示。

```
puppet kick  - d 192 - 168 - 149 - 130 - jfedu.net
#puppet kick  - d 'cat host.txt'
```

```
Debug: /File[/var/lib/puppet/ssl/private_keys]: Autorequiring File[/va
Debug: /File[/var/lib/puppet/state]: Autorequiring File[/var/lib/puppe
Debug: /File[/var/lib/puppet/ssl/certs]: Autorequiring File[/var/lib/p
Debug: /File[/var/lib/puppet/preview]: Autorequiring File[/var/lib/pup
Debug: /File[/var/lib/puppet/ssl]: Autorequiring File[/var/lib/puppet]
Debug: /File[/var/lib/puppet/ssl/certs/192-168-149-128-jfedu.net.pem]:
Debug: /File[/var/lib/puppet/lib]: Autorequiring File[/var/lib/puppet]
Debug: Finishing transaction 70120175045640
Debug: Creating new connection for https://192-168-149-130-jfedu.net:8
Getting status
status is success
192-168-149-130-jfedu.net finished with exit code 0
Finished
```

图 20-26　Puppet 主动通知客户端同步配置

20.12　Puppet 批量部署案例

随着 IT 行业的迅猛发展，传统的运维方式靠大量人力比较吃力，近几年自动化运维管理快速地发展，得到了很多 IT 运维人员的青睐，一个完整的自动化运维包括系统安装、配置管理、服务监控三个方面。以下为 Puppet 应用案例。

某互联网公司新到 100 台硬件服务器，要求统一安装 Linux 系统，并部署上线以及后期的管理配置。对于 Linux 系统安装，需采用批量安装，批量安装系统主流工具为 Kickstart 和 Cobbler，任选其一即可。

如果采用自动安装的话，可以自动初始化系统、内核优化及常见服务、软件客户端等安装。Puppet 客户端可以放在 Kickstart 中安装并配置完毕。

当 Linux 操作系统安装完成后，需要对服务器进行相应的配置，应对高并发网站，例如修改动态 IP 为静态 IP、安装及创建 crontab 任务计划、同步操作系统时间、安装 Zabbix 客户端软件、优化内核参数等，可以基于 Puppet 统一调整。

20.12.1　Puppet 批量修改静态 IP 案例

现需要修改 100 台 Linux 服务器原 DHCP 动态获取的 IP 为 static IP 地址，首先需要修改 IP 脚本，将该脚本推送到客户端，然后执行脚本并重启网卡即可，步骤如下。

（1）修改 IP 为静态 IP 的 shell 脚本代码如下：

```
#!/bin/bash
#auto Change ip netmask gateway scripts
#By author jfedu.net 2017
#Define Path variables
ETHCONF=/etc/sysconfig/network-scripts/ifcfg-eth0
DIR=/data/backup/'date +%Y%m%d'
IPADDR='ifconfig|grep inet|grep 192|head -1|cut -d: -f2|awk '{print $1}''
NETMASK=255.255.255.0
grep dhcp $ETHCONF
if [ $? -eq 0 ];then
        sed -i 's/dhcp/static/g' $ETHCONF
        echo -e "IPADDR=$IPADDR\nNETMASK=$NETMASK\nGATEWAY='echo $IPADDR|awk -F. '
```

```
{print $1"."$2"."$3}''.2" >> $ETHCONF
        echo "The IP configuration success. !"
        service network restart
fi
```

（2）Puppet master 执行 kick 推送配置至 agent 服务器远程，Puppet 客户端修改 IP 脚本代码如下，结果如图 20-27 所示。

```
node default {
file {
        "/tmp/auto_change_ip.sh":
        source =>"puppet://192-168-149-128-jfedu.net/files/auto_change_ip.sh",
        owner => "root",
        group => "root",
        mode => 755,
    }
exec {
        "/tmp/auto_change_ip.sh":
        cwd => "/tmp/",
        user => root,
        path => ["/usr/bin","/usr/sbin","/bin","/bin/sh"],
    }
}
```

```
Debug: /File[/var/lib/puppet/state]: Autorequiring File[/var/lib/pupp
Debug: /File[/var/lib/puppet/ssl/certs]: Autorequiring File[/var/lib/
Debug: /File[/var/lib/puppet/preview]: Autorequiring File[/var/lib/pu
Debug: /File[/var/lib/puppet/ssl]: Autorequiring File[/var/lib/puppet
Debug: /File[/var/lib/puppet/ssl/certs/192-168-149-128-jfedu.net.pem]
Debug: /File[/var/lib/puppet/lib]: Autorequiring File[/var/lib/puppet
Debug: /File[/var/lib/puppet/ssl/private]: Autorequiring File[/var/li
Debug: Finishing transaction 70000861822320
Debug: Creating new connection for https://192-168-149-130-jfedu.net:
Getting status
status is success
192-168-149-130-jfedu.net finished with exit code 0
Finished
[root@192-168-149-128-jfedu ~]#
```

(a) Puppet主动通知客户端同步配置

```
[root@192-168-149-130-jfedu tmp]# cat /etc/sysconfig/network-scrip
DEVICE=eth0
TYPE=Ethernet
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=dhcp
[root@192-168-149-130-jfedu tmp]# cat /etc/sysconfig/network-scrip
DEVICE=eth0
TYPE=Ethernet
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=static
IPADDR=192.168.149.130
NETMASK=255.255.255.0
GATEWAY=192.168.149.2
[root@192-168-149-130-jfedu tmp]#
```

(b) Puppet客户端IP自动配置为static方式

图 20-27　Puppet 配置 static IP

20.12.2　Puppet 批量配置 NTP 同步服务器

在 100 台 Linux 服务器上配置 crontab 任务，修改 ntpdate 与 ntp 服务端同步时间，操作步骤如下。

(1) Puppet master 上创建客户端 node 配置，可以编写 NTP 模块，使用 class 可以定义模块分组，对不同业务进行分组管理，/etc/puppet/modules/ntp/manifests/init.pp 配置文件代码如下，将原 ntpdate 同步时间从 0 点 0 分改成每 5min 同步一次时间，并且修改原 pool.ntp.org 服务器为本地局域网 NTP 时间服务器的 IP 地址。

```
class ntp {
    Exec { path =>"/bin:/sbin:/bin/sh:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"}
    exec {
    "auto change crontab ntp config":
    command =>"sed -i -e '/ntpdate/s/0/*\/5 /2' -e 's/pool.ntp.org/10.1.1.21/' /var/
        spool/cron/root",
    }
}
```

(2) 在/etc/puppet/manifests 目录创建两个文件，分别为 modules.pp 和 nodes.pp，模块入口文件以及 node 配置段。

modules.pp 配置文件内容如下：

```
import "ntp"
```

nodes.pp 配置文件内容如下：

```
node default{
    include ntp
}
```

(3) 在 site.pp 中加载导入 modules.pp 和 nodes.pp 名称，site.pp 代码如下：

```
import "modules.pp"
import "nodes.pp"
```

(4) Puppet master 执行 kick 推送配置至 agent 服务器远程，Puppet 客户端最终结果如图 20-28 所示。

当服务器分组之后，为了更好地管理和配置，可以使用正则表达式来进行定义 node，在定义一个 node 节点时，要指定节点的名字，并使用单引号将名字引起来，然后在大括号中指定需要应用的配置。

客户端节点名字可以是主机名也可以是客户端的正式域名，目前 Puppet 版本还不能使用通配符来指定节点，例如不能用 *.jfedu.net，可以使用正则表达式，代码如下：

```
[root@192-168-149-128-jfedu manifests]# ls
modules.pp  nodes.pp  site.pp
[root@192-168-149-128-jfedu manifests]#
[root@192-168-149-128-jfedu manifests]# cat *
import  "ntp"
node  default {
        include  ntp
}
import  "modules.pp"
import  "nodes.pp"
[root@192-168-149-128-jfedu manifests]# cat /etc/puppet/modules/ntp/mani
class  ntp {
Exec { path =>"/bin:/sbin:/bin/sh:/usr/bin:/usr/sbin:/usr/local/bin:/usr

exec {
        "auto change crontab ntp config":
        command =>"sed -i -e '/ntpdate/s/0/*\/5 /2' -e 's/pool.ntp.org/1
```

(a) Puppet服务端class模块配置

```
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]# crontab -l
0 0  * * *  /usr/sbin/ntpdate  pool.ntp.org >/tmp/ntp.log 2>&1
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]# crontab -l
0 */5   * * *  /usr/sbin/ntpdate  10.1.1.21 >/tmp/ntp.log 2>&1
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]# crontab -l
0 */5   * * *  /usr/sbin/ntpdate  10.1.1.21 >/tmp/ntp.log 2>&1
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]#
[root@192-168-149-130-jfedu ~]#
```

(b) Puppet主动通知客户端修改NTP同步配置

图 20-28 Puppet 配置 NTP

```
node /^Beijing-IDC-web0\d+\-jfedu\.net {
    include ntp
}
```

以上规则会匹配所有在 jfedu.net 域并且主机名以 Beijing-IDC 开头，紧跟 web01，web02，web03，…，web100，…等节点，由此可以进行批量服务器的分组管理。

20.12.3 Puppet 自动部署及同步网站

企业生产环境 100 台服务器，所有服务器要求数据一致，可以采用 rsync 同步，配置 rsync 服务器端，客户端执行脚本命令即可，同样可以使用 Puppet＋shell 脚本来同步，这样比较快捷，也可以使用 Puppet rsync 模块。

（1）Puppet 服务器端配置，/etc/puppet/modules/www/manifests/init.pp 代码如下：

```
class www {
Exec { path =>"/bin:/sbin:/bin/sh:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"}
file {
"/data/sh/rsync_www_client.sh":
source =>"puppet://192-9-117-162-tdt.com/files/www/rsync_www_client.sh",
owner =>"root",
```

```
group =>"root",
mode =>"755",
}
file {
"/etc/rsync.pas":
source =>"puppet://192-9-117-162-tdt.com/files/www/rsync.pas",
owner =>"root",
group =>"root",
mode =>"600",
}
exec {
"auto backup www data":
command =>"mkdir -p /data/backup/'date +%Y%m%d'; mv /data/index /data/backup/www/'date
+%Y%m%d' ; /bin/sh /data/sh/rsync_www_client.sh ",
user =>"root",
subscribe => File["/data/sh/rsync_www_client.sh"],
refreshonly =>"true",
 }
}
```

(2) 在/etc/puppet/manifests 目录创建两个文件，分别为 modules.pp 和 nodes.pp，模块入口文件以及 node 配置段。

modules.pp 配置文件内容如下：

```
import "www"
```

nodes.pp 配置文件内容如下：

```
node /^Beijing-IDC-web0\d+\-jfedu\.net {
include www
}
```

(3) 在 site.pp 中加载导入 modules.pp 和 nodes.pp 名称，site.pp 代码如下：

```
import "modules.pp"
import "nodes.pp"
```

Puppet master 端批量执行通知客户端来同步配置，命令如下：

```
puppet kick -d --host  'cat hosts.txt'
```

(4) cat hosts.txt 内容为需要同步的客户端的主机名，内容如下：

```
Beijing-IDC-web01-jfedu.net
Beijing-IDC-web02-jfedu.net
Beijing-IDC-web03-jfedu.net
Beijing-IDC-web04-jfedu.net
```

第 21 章

Ansible 自动运维企业实战



随着互联网 IT 运维飞速发展，目前市场上涌现了大量的自动化配置维护工具，例如 PSSH、Puppet、Chef、SaltStack、Ansible 等。目前互联网企业使用最多的三款自动化配置工具为 Puppet、Ansible 和 SaltStack。自动配置工具存在的初衷就是为了更方便、快捷地进行配置管理，它易于安装和使用，语法也非常简单易学。

本章向读者介绍 Ansible 工作原理、Ansible 安装配置、生产环境模块讲解、Ansible 企业场景案例、PlayBook 剧本实战及 Ansible 性能调优等内容。

21.1 自动化运维工具简介

曾有媒体报道，Facebook 一个运维人员管理上万台服务器，如果使用手工的方法去维护是很难做到的，基于自动化工具就可以轻松地实现管理上万台、甚至十万台服务器。

下面将介绍 IT 运维主流自动化管理工具 Puppet、SaltStack、Ansible 各自优缺点。

21.1.1 Puppet 自动运维工具特点

Puppet 是早期的 Linux 自动化运维工具，是一种 Linux、UNIX、Windows 平台的集中配置管理系统，发展至今目前已经非常成熟，可以批量管理远程服务器，模块丰富，配置复杂，基于 Ruby 语言编写。最典型的 C/S 模式，需要安装服务端与客户端。

Puppet 采用 C/S 星状的结构，所有的客户端和一个或几个服务器交互，每个客户端周期地（默认半个小时）向服务器发送请求，获得其最新的配置信息，保证和该配置信息同步。

每个 Puppet 客户端每半小时（可以设置）连接一次服务器端，下载最新的配置文件，并且严格按照配置文件来配置客户端。配置完成以后，Puppet 客户端可以反馈给服务器端一个消息，如果出错也会给服务器端反馈一个消息。

Puppet 适用于服务器管理的整个过程，比如初始安装、配置、更新以及系统下线。

21.1.2 SaltStack 自动运维工具特点

SaltStack 与 Puppet 均是 C/S 模式，需安装服务端与客户端，基于 Python 编写，加入 MQ 消息同步，可以使执行命令和执行结果高效返回，但其执行过程需等待客户端全部返回，如果客户端未及时返回或未响应的话，可能会导致部分机器没有执行结果。

21.1.3 Ansible 自动运维工具特点

Ansible 与 SaltStack 均是基于 Python 语言开发，Ansible 只需要在一台普通的服务器上运行即可，不需要在客户端服务器上安装客户端。因为 Ansible 是基于 SSH 远程管理，而 Linux 服务器大都离不开 SSH，所以 Ansible 不需要为配置工作添加额外的支持。

Ansible 安装使用非常简单，而且基于上千个插件和模块，实现各种软件、平台、版本的管理，支持虚拟容器多层级的部署。很多读者在使用 Ansible 工具时，认为 Ansible 比 SaltStatck 执行效率慢，其实不是软件本身慢，是由于 SSH 服务慢，可以优化 SSH 连接速度及使用 Ansible 加速模块，满足企业上万台服务器的维护和管理。

21.2 Ansible 运维工具原理

Ansible 是一款极为灵活的开源工具套件，能够大大简化 UNIX 管理员的自动化配置管理与流程控制方式。它利用推送方式对客户系统加以配置，这样所有工作都可在主服务器端完成。其命令行机制同样非常强大，允许大家利用商业许可 Web UI 实现授权管理与配置，可以通过命令行或者 GUI 来使用 Ansible。运行 Ansible 的服务器这里俗称“管理节点”，通过 Ansible 进行管理的服务器俗称“受控节点”。权威媒体报道，Ansible 于 2015 年被 Red Hat 公司以 1.5 亿美元收购，新版 Red Hat 内置 Ansible 软件。

本书以 Ansible 为案例，基于 Ansible 构建企业自动化运维平台，实现大规模服务器的快速管理和部署。Ansible 将平常复杂的配置工作变得简单，变得更加标准化，也就更容易控制。

Ansible 自动运维管理工具优点如下：

- 轻量级，更新时只需要在操作机上进行一次更新即可；
- 采用 SSH 协议；
- 不需要客户端安装 agent；
- 批量任务执行可以写成脚本，而且不用分发到远程客户端；
- 使用 Python 编写，维护更简单；
- 支持 sudo 普通用户命令；
- 去中心化管理。

Ansible 自动运维管理工具工作原理拓扑如图 21-1 所示。

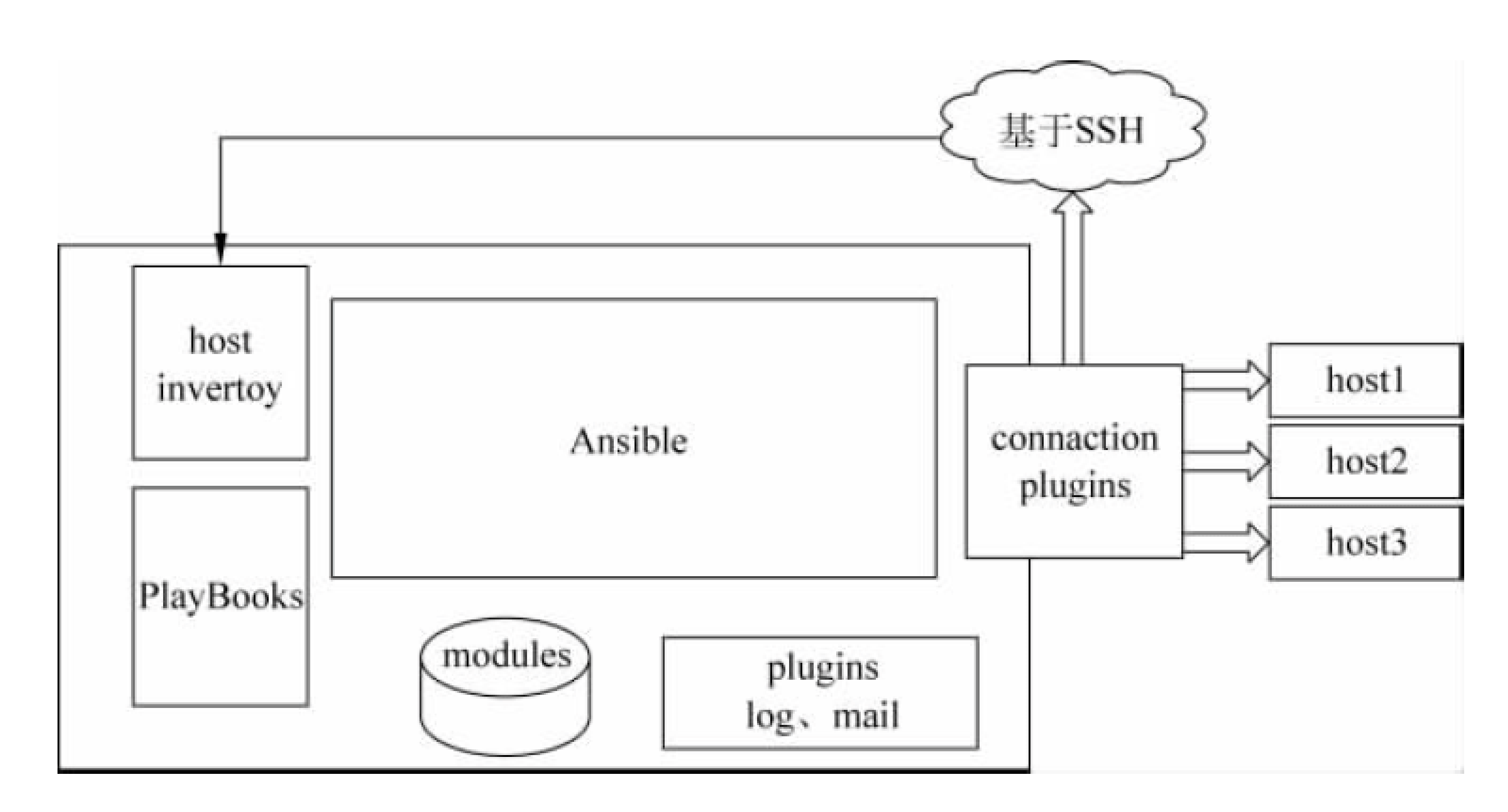


图 21-1　Ansible 工作原理图

21.3　Ansible 管理工具安装配置

Ansible 可以工作在 Linux、BSD、Mac OS X 等平台，对 Python 环境的版本最低要求为 Python 2.6 以上，如果操作系统 Python 软件版本为 2.4，需要升级方可使用 Ansible 工具。

Red Hat、CentOS 操作系统可以直接基于 YUM 工具自动安装 Ansible，CentOS 6.X 或者 CentOS 7.X 安装前，需先安装 epel 扩展源，代码如下：

```
rpm -Uvh http://mirrors.ustc.edu.cn/fedora/epel/6/x86_64/epel-release-6-8.noarch.rpm
yum   install   epel-release  -y
yum   install   ansible       -y
```

Ansible 工具默认主目录为/etc/ansible/，其中 hosts 文件为被管理机 IP 或者主机名列表，ansible.cfg 为 ansible 主配置文件，roles 为角色或者插件路径，默认该目录为空，如图 21-2 所示。

```
[root@localhost ansible]# ls
ansible.cfg  hosts  roles
[root@localhost ansible]# ll
total 28
-rw-r--r--. 1 root root 17718 May 22 20:59 ansible.cfg
-rw-r--r--. 1 root root   410 May 22 20:17 hosts
drwxr-xr-x. 2 root root  4096 Apr 20 05:08 roles
[root@localhost ansible]#
[root@localhost ansible]#
[root@localhost ansible]# pwd
/etc/ansible
[root@localhost ansible]#
[root@localhost ansible]# ls
ansible.cfg  hosts  roles
[root@localhost ansible]#
```

图 21-2　Ansible 主目录信息

Ansible 远程批量管理，其中执行命令是通过 Ad-Hoc 来完成，也即点对点执行命令，能够快速执行，而且不需要保存执行的命令。默认 hosts 文件配置主机列表，可以配置分组，

可以定义各种 IP 及规则，hosts 列表默认配置如图 21-3 所示。

```
# This is the default ansible 'hosts' file.
## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110

## www[001:006].example.com

# Ex 3: A collection of database servers in the 'dbservers' group
```

图 21-3　hosts 主机列表文件内容

Ansible 基于多模块管理，常用的 Ansible 工具管理模块包括：command、shell、script、yum、copy、file、async、docker、cron、mysql_user、ping、sysctl、user、acl、add_host、easy_install、haproxy 等。可以使用 ansible-doc -l|more 查看 Ansible 支持的模块，也可以查看每个模块的帮助文档，用法为 ansible-doc module_name，如图 21-4 所示。

```
[root@www ~]# ansible-doc docker
> DOCKER    (/usr/lib/python2.7/site-packages/ansible/modules/cloud/doc

  This is the original Ansible module for managing the Docker container
  Additional and newer modules are available. For the latest on orchest
  Ansible visit our Getting Started with Docker Guide at
  https://github.com/ansible/ansible/blob/devel/docsite/rst/guide_docke

DEPRECATED:
In 2.2 use M(docker_container) and M(docker_image) instead.

Options (= is mandatory):

- cap_add
        Add capabilities for the container. Requires docker-py >= 0.5.0
        [Default: False]
```

图 21-4　ansible-doc docker 帮助信息

21.4　Ansible 工具参数详解

基于 Ansible 批量管理，需将被管理的服务器 IP 列表添加至/etc/ansible/hosts 文件中，如图 21-5 添加 4 台被管理端 IP 地址，分成 Web 和 DB 两组，本机也可以是被管理机。

```
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups
[web]
192.168.149.128
192.168.149.129
[db]
192.168.149.130
192.168.149.131
```

图 21-5　Ansible hosts 主机列表

Ansible 自动运维工具管理客户端案例操作，由于 Ansible 管理远程服务器基于 SSH，在登录远程服务器执行命令时需要远程服务器的用户名和密码，也可以加入-k 参数手动输入密码或者基于 ssh-keygen 生成免秘钥。

Ansible 自动化批量管理工具主要参数详解如下：

- -v,-verbose：打印详细模式。
- -i PATH,-inventory=PATH：指定 host 文件路径。
- -f NUM,-forks=NUM：指定 fork 开启同步进程的个数，默认为 5。
- -m NAME,-module-name=NAME：指定 module 名称，默认模块为 command。
- -a MODULE_ARGS：module 模块的参数或者命令。
- -k,-ask-pass：输入远程被管理端密码。
- -sudo：基于 sudo 用户执行。
- -K,-ask-sudo-pass：提示输入 sudo 密码与 sudo 一起使用。
- -u USERNAME,-user=USERNAME：指定执行用户。
- -C,--check：测试执行过程，不改变真实内容，相当于预演。
- -T TIMEOUT：执行命令超时时间，默认为 10s。
- --version：查看 Ansible 软件版本信息。

21.5 Ansible ping 模块实战

Ansible 最基础的模块为 ping 模块，主要用于判断远程客户端是否在线，用于 ping 本身服务器，返回值为 changed、ping。

Ansible ping 模块企业常用案例如下：

使用 Ansible ping 服务器状态，代码如下，结果如图 21-6 所示。

```
ansible -k all -m ping
```

```
[root@localhost ~]# ansible -k all  -m ping
SSH password:
192.168.149.129 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.149.130 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.149.131 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.149.128 | SUCCESS => {
```

图 21-6　Ansible ping 服务器状态

21.6 Ansible command 模块实战

Ansible command 模块为 Ansible 默认模块，主要用于执行 Linux 基础命令，可以执行远程服务器命令执行、任务执行等操作。command 模块使用详解如下：

- Chdir：执行命令前，切换到目录。
- Creates：当该文件存在时，则不执行该步骤。
- Executable：换用 shell 环境执行命令。
- Free_form：需要执行的脚本。
- Removes：当该文件不存在时，则不执行该步骤。
- Warn：若在 ansible.cfg 中存在告警，如果设定了 false，不会警告此行。

Ansible command 模块企业常用案例如下。

(1) Ansible command 模块远程执行 date 命令，代码如下，执行结果如图 21-7 所示。

```
ansible -k -i /etc/ansible/hosts all -m command -a "date"
```

```
[root@localhost ansible]# ansible -k -i /etc/ansible/hosts  all  -m co
SSH password:
192.168.149.128 | SUCCESS | rc=0 >>
Wed May 31 00:24:43 CST 2017

192.168.149.131 | SUCCESS | rc=0 >>
Sat May 20 09:34:37 CST 2017

192.168.149.130 | SUCCESS | rc=0 >>
Wed May 24 21:47:33 CST 2017

192.168.149.129 | SUCCESS | rc=0 >>
Wed May 31 00:25:09 CST 2017

[root@localhost ansible]#
```

图 21-7　Ansible command date 命令执行结果

(2) Ansible command 模块远程执行 ping 命令，代码如下，执行结果如图 21-8 所示。

```
ansible -k all -m command -a "ping -c 1 www.baidu.com"
```

```
[root@localhost ansible]# ansible -k all  -m command -a  "ping -c 1 w
SSH password:
192.168.149.129 | SUCCESS | rc=0 >>
PING www.a.shifen.com (119.75.218.70) 56(84) bytes of data.
64 bytes from 119.75.218.70: icmp_seq=1 ttl=128 time=6.49 ms

--- www.a.shifen.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 12ms
rtt min/avg/max/mdev = 6.498/6.498/6.498/0.000 ms

192.168.149.130 | SUCCESS | rc=0 >>
PING www.a.shifen.com (119.75.217.109) 56(84) bytes of data.
64 bytes from 119.75.217.109: icmp_seq=1 ttl=128 time=7.25 ms

--- www.a.shifen.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 22ms
rtt min/avg/max/mdev = 7.250/7.250/7.250/0.000 ms
```

图 21-8　Ansible command ping 命令执行结果

（3）Ansible hosts 正则模式远程执行 df -h，代码如下，执行结果如图 21-9 所示。

```
ansible -k 192.168.149.13* -m command -a "df -h"
```

```
[root@localhost ansible]# ansible -k  192.168.149.13*  -m  command
SSH password:
192.168.149.131 | SUCCESS | rc=0 >>
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        30G   15G   14G  52% /
tmpfs           242M     0  242M   0% /dev/shm
/dev/sda1       194M  190M     0 100% /boot

192.168.149.130 | SUCCESS | rc=0 >>
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        30G   16G   12G  57% /
tmpfs           242M     0  242M   0% /dev/shm
/dev/sda1       194M   27M  158M  15% /boot

[root@localhost ansible]#
```

图 21-9 Ansible command df -h 命令执行结果

21.7 Ansible copy 模块实战

Ansible copy 模块主要用于文件或者目录复制，支持文件、目录、权限、用户组功能，copy 模块使用详解如下：

- src：Ansible 端源文件或者目录，空文件夹不复制。
- content：用来替代 src，用于将指定文件的内容复制到远程文件内。
- dest：客户端目标目录或者文件，需要绝对路径。
- backup：复制之前，先备份远程节点上的原始文件。
- directory_mode：用于复制文件夹，新建的文件会被复制，而老旧的不会被复制。
- follow：支持 link 文件复制。
- force：覆盖远程主机不一致的内容。
- group：设定远程主机文件夹的组名。
- mode：指定远程主机文件及文件夹的权限。
- owner：设定远程主机文件夹的用户名。

Ansible copy 模块企业常用案例如下。

（1）Ansible copy 模块操作，src 表示源文件，dest 表示目标目录或者文件，owner 指定拥有者，代码如下，执行结果如图 21-10 所示。

```
ansible -k all -m copy -a 'src=/etc/passwd dest=/tmp/ mode=755 owner=root'
```

（2）Ansible copy 模块操作，content 表示文件内容，dest 表示目标文件，owner 指定拥有者，代码如下，执行结果如图 21-11 所示。

```
ansible -k all -m copy -a 'content="Hello World" dest=/tmp/jfedu.txt mode=755 owner=
root'
```

```
[root@localhost ~]# ansible -k all -m copy -a 'src=/etc/passwd  dest=
SSH password:
192.168.149.131 | SUCCESS => {
    "changed": true,
    "checksum": "6968f8053525cb8a821b3855d1860ad39f8f0e5d",
    "dest": "/tmp/passwd",
    "gid": 0,
    "group": "root",
    "md5sum": "d65ee7c1ff1b9868a6ee72eaf8666b03",
    "mode": "0755",
    "owner": "root",
    "size": 1791,
    "src": "/root/.ansible/tmp/ansible-tmp-1496163407.29-279065359298
    "state": "file",
    "uid": 0
```

图 21-10 Ansible copy 复制文件

```
root'
SSH password:
192.168.149.129 | SUCCESS => {
    "changed": true,
    "checksum": "7b502c3a1f48c8609ae212cdfb639dee39673f5e",
    "dest": "/tmp/jfedu.txt",
    "gid": 0,
    "group": "root",
    "md5sum": "3e25960a79dbc69b674cd4ec67a72c62",
    "mode": "0755",
    "owner": "root",
    "size": 11,
    "src": "/root/.ansible/tmp/ansible-tmp-1496167669.85-1119688532
    "state": "file",
    "uid": 0
```

图 21-11 Ansible copy 追加内容

（3）Ansible copy 模块操作，content 表示文件内容，dest 表示目标文件，owner 指定拥有者，backup＝yes 开启备份，代码如下，执行结果如图 21-12 所示。

```
ansible -k all -m copy -a 'content="Hello World" dest=/tmp/jfedu.txt backup=yes mode=
755 owner=root'
```

```
jfedu-net-129 tmp]# ls
xADO  ansible_YOWbvE  cacti.log  httpd-2.2.31.tar.gz  jfedu.txt  jfedu.
jfedu-net-129 tmp]#
jfedu-net-129 tmp]# ll

 2 root root    4096 May 31 01:39 ansible_G2xADO
 2 root root    4096 May 31 01:39 ansible_YOWbvE
 1 root root    1064 May 31 01:35 cacti.log
 1 root root 4711452 May 31 01:22 httpd-2.2.31.tar.gz
 1 root root      16 May 31 02:03 jfedu.txt
 1 root root      11 May 31 02:01 jfedu.txt.28886.2017-05-31@02:03:38~
 3 root root    4096 May 31 01:36 test
jfedu-net-129 tmp]#
jfedu-net-129 tmp]# ll jfedu.txt.28886.2017-05-31\@02\:03\:38~
 1 root root 11 May 31 02:01 jfedu.txt.28886.2017-05-31@02:03:38~
```

图 21-12 Ansible copy 客户端备份结果

21.8 Ansible YUM 模块实战

Ansible YUM 模块主要用于软件的安装、升级、卸载，支持红帽 rpm 软件包的管理，YUM 模块使用详解如下：

- conf_file：设定远程 YUM 执行时所依赖的 YUM 配置文件。
- disable_gpg_check：安装软件包之前是否检查 gpg key。
- name：需要安装的软件名称，支持软件组安装。
- update_cache：安装软件前更新缓存。
- enablerepo：指定 repo 源名称。
- skip_broken：跳过异常软件节点。
- state：软件包状态，包括 installed、present、latest、absent、removed。

Ansible YUM 模块企业常用案例如下。

（1）Ansible YUM 模块操作，name 表示需安装的软件名称，state 表示状态，常见 state= installed 表示安装软件，代码如下，执行结果如图 21-13 所示。

```
ansible all -k -m yum -a "name=sysstat,screen state=installed"
```

```
    "msg": "",
    "rc": 0,
    "results": [
        "sysstat-9.0.4-33.el6.x86_64 providing sysstat is already
        "screen-4.0.3-19.el6.x86_64 providing screen is already i
    ]
}
192.168.149.130 | SUCCESS => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "screen-4.0.3-19.el6.x86_64 providing screen is already i
        "Loaded plugins: fastestmirror\nLoading mirror speeds fro
irror.lzu.edu.cn\n * extras: mirror.bit.edu.cn\n * updates: mirro
```

图 21-13　Ansible YUM 安装软件包

（2）Ansible YUM 模块操作，name 表示需安装的软件名称，state 表示状态，常见 state= absent 表示卸载软件，代码如下，执行结果如图 21-14 所示。

```
ansible all -k -m yum -a "name=sysstat,screen state=absent"
```

```
SSH password:
192.168.149.129 | SUCCESS => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "Loaded plugins: fastestmirror\nSetting up Remove Process
n---> Package screen.x86_64 0:4.0.3-19.el6 will be erased\n---> P
 Finished Dependency Resolution\n\nDependencies Resolved\n\n=====
==================\n Package          Arch            Version
=============================================================\nRe
           @base          795 k\n sysstat          x86_64
saction Summary\n================================================
e(s)\n\nInstalled size: 1.6 M\nDownloading Packages:\nRunning rpm
 Succeeded\nRunning Transaction\n\r  Erasing    : screen-4.0.3-19
```

图 21-14　Ansible YUM 卸载软件包

（3）Ansible YUM 模块操作，name 表示需安装的软件名称，state 表示状态，常见 state= installed 表示安装软件，disable_gpg_check=no 表示不检查 key，代码如下，执行结果如图 21-15 所示。

```
ansible 192.168.149.129 -k -m yum -a "name=sysstat,screen state=installed disable_gpg_
check=no"
```

```
SSH password:
192.168.149.129 | SUCCESS => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "Loaded plugins: fastestmirror\nLoading mirror speeds fro
 mirror.bit.edu.cn\n * updates: mirror.bit.edu.cn\nSetting up Ins
ction check\n---> Package screen.x86_64 0:4.0.3-19.el6 will be in
 be installed\n--> Finished Dependency Resolution\n\nDependencies
====================================\n Package              Arch
==========================================================
        4.0.3-19.el6           base            494 k\n sysstat
      234 k\n\nTransaction Summary\n============================
stall        2 Package(s)\n\nTotal download size: 729 k\nInstalled
```

图 21-15　Ansible YUM 安装软件包，不检查 key

21.9　Ansible file 模块实战

Ansible file 模块主要用于对文件的创建、删除、修改、权限、属性的维护和管理，file 模块使用详解如下：

- src：Ansible 端源文件或者目录。
- follow：支持 link 文件复制。
- force：覆盖远程主机不一致的内容。
- group：设定远程主机文件夹的组名。
- mode：指定远程主机文件及文件夹的权限。
- owner：设定远程主机文件夹的用户名。
- path：目标路径，也可以用 dest，name 代替。
- state：状态包括 file、link、directory、hard、touch、absent。
- attributes：文件或者目录特殊属性。

Ansible file 模块企业常用案例如下。

（1）Ansible file 模块操作，path 表示目录的名称和路径，state=directory 表示创建目录，代码如下，执行结果如图 21-16 所示。

```
ansible -k 192.168.* -m file -a "path=/tmp/'date +%F' state=directory mode=755"
```

```
[root@localhost ~]# ansible -k 192.168.* -m  file  -a  "path=/tmp/test
SSH password:
192.168.149.129 | SUCCESS => {
    "changed": true,
    "dest": "/tmp/test.txt",
    "gid": 0,
    "group": "root",
    "mode": "0644",
    "owner": "root",
    "size": 6,
    "state": "file",
    "uid": 0
}
192.168.149.130 | SUCCESS => {
    "changed": true,
```

图 21-16　Ansible file 创建目录

（2）Ansible file 模块操作，path 表示目录的名称和路径，state=touch 表示创建文件，代码如下，执行结果如图 21-17 所示。

```
ansible -k 192.168.* -m file -a "path=/tmp/jfedu.txt state=touch mode=755"
```

```
[root@localhost ~]# ansible -k 192.168.* -m file -a "path=/tmp
SSH password:
192.168.149.130 | SUCCESS => {
    "changed": true,
    "dest": "/tmp/jfedu.txt",
    "gid": 0,
    "group": "root",
    "mode": "0644",
    "owner": "root",
    "size": 0,
    "state": "file",
    "uid": 0
}
192.168.149.131 | SUCCESS => {
    "changed": true,
```

图 21-17　Ansible file 创建文件

21.10　Ansible user 模块实战

Ansible user 模块主要用于操作系统用户、组、权限、密码等操作，user 模块使用详解如下：

- system：默认创建为普通用户，为 yes 则创建系统用户。
- append：添加一个新的组。
- comment：新增描述信息。
- createhome：给用户创建家目录。
- force：强制删除用户。
- group：创建用户主组。
- groups：将用户加入组或者附属组添加。
- home：指定用户的家目录。
- name：表示状态，是否 create、remove、modify。
- password：指定用户的密码，此处为加密密码。
- remove：删除用户。
- shell：设置用户的 shell 登录环境。
- uid：设置用户 ID。
- update_password：修改用户密码。
- state：用户状态，默认为 present，表示新建用户。

Ansible user 模块企业常用案例如下。

（1）Ansible user 模块操作，name 表示用户名称，home 表示其家目录，代码如下，执行结果如图 21-18 所示。

```
ansible  -k  192.168.149.*   -m  user  -a  "name=jfedu  home=/tmp/"
```

```
[root@localhost ~]# ansible  -k  192.168.149.*  -m  user  -a
SSH password:
192.168.149.129 | SUCCESS => {
    "changed": true,
    "comment": "",
    "createhome": true,
    "group": 504,
    "home": "/tmp/",
    "name": "jfedu",
    "shell": "/bin/bash",
    "state": "present",
    "stderr": "useradd: warning: the home directory already exist
,
    "stderr_lines": [
        "useradd: warning: the home directory already exists.",
```

图 21-18　Ansible user 创建新用户

（2）Ansible user 模块操作，name 表示用户名称，home 表示家目录并且指定其 shell，代码如下，执行结果如图 21-19 所示。

```
ansible  -k  192.168.149.*   -m  user  -a  "name=jfedu  home=/tmp/ shell=/sbin/
nologin"
```

```
192.168.149.131 | SUCCESS => {
    "append": false,
    "changed": false,
    "comment": "",
    "group": 501,
    "home": "/tmp/",
    "move_home": false,
    "name": "jfedu",
    "shell": "/sbin/nologin",
    "state": "present",
    "uid": 501
}
192.168.149.129 | SUCCESS => {
    "changed": true,
    "comment": "",
```

图 21-19　Ansible user 指定 shell 环境

（3）Ansible user 模块操作，name 表示用户名称，state=absent 表示删除用户，代码如下，执行结果如图 21-20 所示。

```
ansible  -k  192.168.149.*   -m  user  -a  "name=jfedu  state=absent force=yes"
```

```
[root@localhost ~]# ansible  -k  192.168.149.*  -m  user  -a
SSH password:
192.168.149.129 | SUCCESS => {
    "changed": true,
    "force": true,
    "name": "jfedu",
    "remove": false,
    "state": "absent"
}
192.168.149.131 | SUCCESS => {
    "changed": true,
    "force": true,
    "name": "jfedu",
    "remove": false,
    "state": "absent"
```

图 21-20　Ansible user 删除用户

21.11 Ansible cron 模块实战

Ansible cron 模块主要用于添加、删除、更新操作系统 crontab 任务计划，cron 模块使用详解如下：

- name：任务计划名称。
- cron_file：替换客户端该用户的任务计划的文件。
- minute：分(0-59，*，*/2)。
- hour：时(0-23，*，*/2)。
- day：日(1-31，*，*/2)。
- month：月(1-12，*，*/2)。
- weekday：周(0-6 或 1-7，*)。
- job：任何计划执行的命令，state 要等于 present。
- backup：是否备份之前的任务计划。
- user：新建任务计划的用户。
- state：指定任务计划 present、absent。

Ansible cron 模块企业常用案例如下。

(1) Ansible cron 模块操作，基于 cron 模块，创建 crontab 任务计划，代码如下，执行结果如图 21-21 所示。

```
ansible -k all -m cron -a "minute=0 hour=0 day=* month=* weekday=* name='Ntpdate
server for sync time'  job='/usr/sbin/ntpdate 139.224.227.121'"
```

```
[root@localhost ~]# ansible -k all -m cron -a "minute=0 ho
 sync time'  job='/usr/sbin/ntpdate 139.224.227.121'"
SSH password:
192.168.149.129 | SUCCESS => {
    "changed": true,
    "envs": [],
    "jobs": [
        "Ntpdate server for sync time"
    ]
}
192.168.149.130 | SUCCESS => {
    "changed": false,
    "envs": [],
    "jobs": [
        "Ntpdate server for sync time"
```

图 21-21 Ansible cron 添加任务计划

(2) Ansible cron 模块操作，基于 cron 模块，备份 crontab 任务计划，backup=yes 表示开启备份，备份文件存放于客户端/tmp/，代码如下，执行结果如图 21-22 所示。

```
ansible -k all -m cron -a "minute=0 hour=0 day=* month=* weekday=* name='Ntpdate
server for sync time' backup=yes job='/usr/sbin/ntpdate pool.ntp.org'"
```

(3) Ansible cron 模块操作，基于 cron 模块，删除 crontab 任务计划，代码如下，执行结

```
[root@localhost ~]# ansible -k all -m cron -a "minute=0 ho
 sync time' backup=yes job='/usr/sbin/ntpdate pool.ntp.org'"
SSH password:
192.168.149.129 | SUCCESS => {
    "backup_file": "/tmp/crontab_Hp7sX",
    "changed": true,
    "envs": [],
    "jobs": [
        "Ntpdate server for sync time"
    ]
}
192.168.149.130 | SUCCESS => {
    "backup_file": "/tmp/crontabjzWpij",
    "changed": true,
    "envs": [],
```

图 21-22 Ansible cron 创建任务计划

果如图 21-23 所示。

```
ansible -k all -m cron -a "name='Ntpdate server for sync time' state=absent"
```

```
[root@localhost ~]# ansible -k all -m cron -a "name='Ntpdat
SSH password:
192.168.149.130 | SUCCESS => {
    "changed": true,
    "envs": [],
    "jobs": []
}
192.168.149.129 | SUCCESS => {
    "changed": true,
    "envs": [],
    "jobs": []
}
192.168.149.131 | SUCCESS => {
    "changed": true,
    "envs": [],
```

图 21-23 Ansible cron 删除任务计划

21.12 Ansible synchronize 模块实战

Ansible synchronize 模块主要用于目录、文件同步，主要基于 rsync 命令工具同步目录和文件，synchronize 模块使用详解如下：

- compress：开启压缩，默认为开启。
- archive：是否采用归档模式同步，保证源和目标文件属性一致。
- checksum：是否效验。
- dirs：以非递归的方式传输目录。
- links：同步链接文件。
- recursive：是否递归 yes/no。
- rsync_opts：使用 rsync 的参数。
- copy_links：同步的时候是否复制链接。
- delete：删除源中没有而目标存在的文件。
- src：源目录及文件。

- dest：目标目录及文件。
- dest_port：目标接受的端口。
- rsync_path：服务的路径，指定 rsync 命令来在远程服务器上运行。
- rsync_timeout：指定 rsync 操作的 IP 超时时间。
- set_remote_user：设置远程用户名。
- --exclude=.log：忽略同步.log 结尾的文件。
- mode：同步的模式，rsync 同步的方式 push、pull，默认都是推送 push。

Ansible synchronize 模块企业常用案例如下。

(1) Ansible synchronize 模块操作，src 为源目录，dest 为目标目录，代码如下，执行结果如图 21-24 所示。

```
ansible -k all -m synchronize -a 'src=/tmp/ dest=/tmp/'
```

```
192.168.149.129 | SUCCESS => {
    "changed": true,
    "cmd": "/usr/bin/rsync --delay-updates -F --compress --archiv
o StrictHostKeyChecking=no --out-format=<<CHANGED>>%i %n%L /tmp/
    "msg": ".d..t...... ./\n<f.st...... cacti.log\n<f+++++++++ cr
.txt\n<f..t...... test.txt\n<f+++++++++ tmp41pHom\n<f+++++++++ tm
\n<f+++++++++ tmpbKomV7\ncd+++++++++ ansible_RTiOm9/\n<f+++++++++
.zip\n<f+++++++++ ansible_RTiOm9/ansible_module_synchronize.py\n"
    "rc": 0,
    "stdout_lines": [
        ".d..t...... ./",
        "<f.st...... cacti.log",
        "<f+++++++++ crontabSqrvcV",
        "<f.st...... jfedu.txt",
```

图 21-24 Ansible 目录同步

(2) Ansible synchronize 模块操作，src 为源目录，dest 为目标目录、compress=yes 表示开启压缩，delete=yes 表示数据一致，rsync_opts 为同步参数，--exclude 表示排除文件，代码如下，执行结果如图 21-25 所示。

```
ansible -k all -m synchronize -a 'src=/tmp/ dest=/tmp/ compress=yes delete=yes rsync_
opts=--no-motd,--exclude=.txt'
```

```
192.168.149.130 | SUCCESS => {
    "changed": true,
    "cmd": "/usr/bin/rsync --delay-updates -F --compress --delet
n/ssh -S none -o StrictHostKeyChecking=no --no-motd --exclude=.t
n%L /tmp/ 192.168.149.130:/tmp/",
    "msg": ".d..t...... ./\n<f.st...... cacti.log\n<f+++++++++ c
.txt\n<f..t...... test.txt\n<f+++++++++ tmp41pHom\n<f+++++++++ t
\n<f+++++++++ tmpbKomV7\ncd+++++++++ ansible_CKVg5Y/\n<f+++++++++
.zip\n<f+++++++++ ansible_CKVg5Y/ansible_module_synchronize.py\n
+++++++++ ansible_KGnLPC/ansible_modlib.zip\n<f+++++++++ ansible
ize.py\ncd+++++++++ ansible_L_uQYR/\n<f+++++++++ ansible_L_uQYR/
 ansible_L_uQYR/ansible_module_synchronize.py\ncd+++++++++ ansib
_zghbN7/ansible_modlib.zip\n<f+++++++++ ansible_zghbN7/ansible_m
   ansible_cT1GmK/ansible_module_synchronize.py\n*deleting    ans
```

图 21-25 Ansible 目录同步排除.txt 文件

21.13　Ansible shell 模块实战

Ansible shell 模块主要用于远程客户端上执行各种 shell 命令或者运行脚本，远程执行命令通过/bin/sh 环境来执行，支持比 command 更多的指令，shell 模块使用详解如下：

- Chdir：执行命令前，切换到目录。
- Creates：当该文件存在时，则不执行该步骤。
- Executable：换用 shell 环境执行命令。
- Free_form：需要执行的脚本。
- Removes：当该文件不存在时，则不执行该步骤。
- Warn：如果在 ansible.cfg 中存在告警，如果设定了 False，不会警告此行。

Ansible shell 模块企业常用案例如下。

（1）Ansible shell 模块操作，-m shell 指定模块为 shell，远程执行 shell 脚本，远程执行脚本也可采用 script 模块。并把执行结果追加至客户端服务器/tmp/var.log 文件，代码如下，执行结果如图 21-26 所示。

```
ansible -k all -m shell -a "/bin/sh /tmp/variables.sh >>/tmp/var.log"
```

```
[root@localhost sh]# ansible -k all -m shell -a "/bin/sh /
SSH password:
192.168.149.131 | SUCCESS | rc=0 >>

192.168.149.129 | SUCCESS | rc=0 >>

192.168.149.130 | SUCCESS | rc=0 >>

192.168.149.128 | SUCCESS | rc=0 >>
```

图 21-26　Ansible shell 远程执行脚本

（2）Ansible shell 模块操作，远程执行创建目录命令，执行之前切换在/tmp 目录，屏蔽警告信息，代码如下，执行结果如图 21-27 所示。

```
ansible -k all -m shell -a "mkdir -p 'date + %F' chdir=/tmp/ state=directory warn=no"
```

```
[root@localhost sh]# ansible -k all -m shell -a "mkdir
no"
SSH password:
192.168.149.130 | SUCCESS | rc=0 >>

192.168.149.129 | SUCCESS | rc=0 >>

192.168.149.131 | SUCCESS | rc=0 >>

192.168.149.128 | SUCCESS | rc=0 >>
```

图 21-27　Ansible shell 远程执行脚本

(3) Ansible shell 模块操作，-m shell 指定模块为 shell，远程客户端查看 http 进程是否启动，代码如下，执行结果如图 21-28 所示。

```
ansible -k all -m shell -a "ps -ef |grep http"
```

```
ansible -k all -m shell -a "ps -ef |grep http"

CESS | rc=0 >>
 0 May24 ?        00:00:11 /usr/local/zabbix/sbin/zabbix_
77 sec, idle 5 sec]
 0 20:02 pts/0    00:00:00 /bin/sh -c ps -ef |grep http
 0 20:02 pts/0    00:00:00 grep http

CESS | rc=0 >>
 0 May20 ?        00:00:07 /usr/sbin/httpd
 0 03:28 ?        00:00:00 /usr/sbin/httpd
 0 03:28 ?        00:00:00 /usr/sbin/httpd
 0 03:28 ?        00:00:00 /usr/sbin/httpd
```

图 21-28 Ansible shell 远程查看进程

(4) Ansible shell 模块操作，-m shell 指定模块为 shell，远程客户端查看 crontab 任务计划，代码如下，执行结果如图 21-29 所示。

```
ansible   -k   all -m shell -a "crontab -l"
```

```
calhost sh]# ansible -k all -m shell -a "crontab -l"
word:
149.130 | SUCCESS | rc=0 >>

149.131 | SUCCESS | rc=0 >>
 * ntpdate pool.ntp.org >/dev/null 2>&1

149.129 | SUCCESS | rc=0 >>

149.128 | SUCCESS | rc=0 >>
*  *  *  /usr/bin/php /var/www/html/cacti/poller.php >>
```

图 21-29 Ansible shell 远程查看任务计划

21.24 Ansible service 模块实战

Ansible service 模块主要用于远程客户端各种服务管理，包括启动、停止、重启、重新加载等，service 模块使用详解如下：

- enabled：是否开机启动服务。
- name：服务名称。
- runlevel：服务启动级别。
- arguments：服务命令行参数传递。
- state：服务操作状态，状态包括 started、stopped、restarted、reloaded。

Ansible service 模块企业常用案例如下。

(1) Ansible service 模块操作，远程重启 httpd 服务，代码如下，执行结果如图 21-30

所示。

```
ansible -k all -m service -a "name=httpd state=restarted"
```

```
[root@localhost sh]# ansible -k all -m service -a "name=
SSH password:
192.168.149.130 | SUCCESS => {
    "changed": true,
    "name": "httpd",
    "state": "started"
}
192.168.149.128 | SUCCESS => {
    "changed": true,
    "name": "httpd",
    "state": "started"
}
192.168.149.131 | SUCCESS => {
    "changed": true
```

图 21-30 Ansible service 重启 httpd 服务

（2）Ansible service 模块操作，远程重启网卡服务，指定参数 eth0，代码如下，执行结果如图 21-31 所示。

```
ansible -k all -m service -a "name=network args=eth0 state=restarted"
```

```
[root@localhost sh]# ansible -k all  -m  service  -a  "na
SSH password:
192.168.149.129 | SUCCESS => {
    "changed": true,
    "name": "network",
    "state": "started"
}
192.168.149.128 | SUCCESS => {
    "changed": true,
    "name": "network",
    "state": "started"
}
192.168.149.131 | SUCCESS => {
    "changed": true,
```

图 21-31 Ansible service 重启 network 服务

（3）Ansible service 模块操作，远程开机启动 nfs 服务，设置 3，5 级别自动启动，代码如下，执行结果如图 21-32 所示。

```
ansible -k all -m service -a "name=nfs enabled=yes runlevel=3,5"
```

```
[root@localhost sh]# ansible -k all  -m  service  -a  "
SSH password:
192.168.149.130 | SUCCESS => {
    "changed": false,
    "enabled": true,
    "name": "nfs"
}
192.168.149.131 | SUCCESS => {
    "changed": false,
    "enabled": true,
    "name": "nfs"
}
192.168.149.129 | SUCCESS => {
    "changed": true,
```

图 21-32 Ansible service 开机启动 nfs 服务

21.15 Ansible PlayBook 应用

如上使用 Ad-hoc 方式点对点命令执行，可以管理远程主机，如果服务器数量很多，配置信息比较多，还可以利用 Ansible PlayBook 编写剧本，从而以更加简便的方式实现任务处理的自动化与流程化。

PlayBook 是由一个或多个“play”组成的列表，play 的主要功能是为 Ansible 中的 task 定义好的角色，指定剧本对应的服务器组。

从根本上说，task 是一个任务，task 调用 Ansible 各种模块 module，将多个 paly 组织在一个 PlayBook 剧本中，然后组成一个非常完整的流程控制集合。

基于 Ansible PlayBook 还可以收集命令、创建任务集，这样能够大大降低管理工作的复杂程度，PlayBook 采用 YAML 语法结构，易于阅读，方便配置。

YAML(yet another markup language)是一种直观的能够被电脑识别的数据序列化格式，是一个容易阅读，容易和脚本语言交互，用来表达资料序列的编程语言。它参考了其他多种语言，包括 XML、C 语言、Python、Perl 以及电子邮件格式 RFC2822，是类似于标准通用标记语言的子集 XML 的数据描述语言，但语法比 XML 简单很多。

YAML 使用空白字符和分行来分隔资料，适合用 grep、Python、Perl、Ruby 操作。

(1) YAML 语言特性如下：

- 可读性强；
- 和脚本语言的交互性好；
- 使用实现语言的数据类型；
- 一致的信息模型；
- 易于实现；
- 可以基于流来处理；
- 可扩展性强。

(2) PlayBooks 组件包括内容如下：

- target：定义 PlayBook 的远程主机组。
- variable：定义 PlayBook 使用的变量。
- task：定义远程主机上执行的任务列表。
- handler：定义 task 执行完成以后需要调用的任务，例如配置文件被改动，则启动 handler 任务重启相关联的服务。

(3) target 常用参数详解如下：

- hosts：定义远程主机组。
- user：执行该任务的用户。
- sudo：设置为 yes 的时候，执行任务的时候使用 root 权限。

- sudo_user：指定 sudo 普通用户。
- connection：默认基于 SSH 连接客户端。
- gather_facks：获取远程主机 facts 基础信息。

(4) variable 常用参数详解如下：

- vars：定义格式，变量名：变量值。
- vars_files：指定变量文件。
- vars_prompt：用户交互模式自定义变量。
- setup：模块取远程主机的值。

(5) task 常用参数详解如下：

- name：任务显示名称也即屏幕显示信息。
- action：定义执行的动作。
- copy：复制本地文件到远程主机。
- template：复制本地文件到远程主机，可以引用本地变量。
- service：定义服务的状态。

Ansible PlayBook 案例演示如下。

(1) 远程主机安装 Nginx Web 服务，PlayBook 代码如下，执行结果如图 21-33 所示。

```
- hosts: all
  remote_user: root
  tasks:
   - name: Jfedu Pcre-devel and Zlib LIB Install.
     yum: name=pcre-devel,pcre,zlib-devel state=installed
   - name: Jfedu Nginx Web Server Install Process.
     shell: cd /tmp; rm -rf nginx-1.12.0.tar.gz; wget http://nginx.org/download/nginx-1.
12.0.tar.gz; tar x
zf nginx-1.12.0.tar.gz; cd nginx-1.12.0; ./configure --prefix=/usr/local/nginx; make;
make install
```

```
[root@localhost ~]# ansible-playbook nginx_install.yaml

PLAY [all] ************************************************

TASK [Nginx Web Server Rewrite Install] *****************
ok: [192.168.149.129]
ok: [192.168.149.128]

TASK [Jfedu install Nginx Web Server Process] **********
changed: [192.168.149.128]
changed: [192.168.149.129]

PLAY RECAP ************************************************
192.168.149.128            : ok=2    changed=1    unrea
192.168.149.129            : ok=2    changed=1    unrea
```

图 21-33 Ansible PlayBook 远程 Nginx 安装

（2）检测远程主机 Nginx 目录是否存在，不存在则安装 Nginx Web 服务，安装完并启动 Nginx，PlayBook 代码如下，执行结果如图 21-34 所示。

```
- hosts: all
  remote_user: root
  tasks:
      - name: Nginx server Install 2017
        file: path=/usr/local/nginx/ state=directory
        notify:
            - nginx install
            - nginx start
  handlers:
      - name: nginx install
        shell: cd /tmp; rm -rf nginx-1.12.0.tar.gz; wget http://nginx.org/download/nginx-
1.12.0.tar.gz; tar xzf nginx-1.12.0
.tar.gz; cd nginx-1.12.0; ./configure --prefix=/usr/local/nginx; make; make install
      - name: nginx start
        shell: /usr/local/nginx/sbin/nginx
```

```
[root@localhost ~]# ansible-playbook nginx.yaml
PLAY [all] ************************************************************

TASK [Nginx server Install 2017] ***************************************
changed: [192.168.149.129]

RUNNING HANDLER [nginx install] ****************************************
changed: [192.168.149.129]

RUNNING HANDLER [nginx start] ******************************************
changed: [192.168.149.129]

PLAY RECAP ************************************************************
192.168.149.129            : ok=3    changed=3    unreachable=0
```

图 21-34　Ansible PlayBook Nginx 触发安装

（3）检测远程主机内核参数配置文件是否更新，如果更新则执行命令 sysctl -p 使内核参数生效，PlayBook 代码如下，执行结果如图 21-35 所示。

```
- hosts: all
  remote_user: root
  tasks:
      - name: Linux kernel config 2017
        copy: src=/data/sh/sysctl.conf dest=/etc/
        notify:
            - source sysctl
  handlers:
      - name: source sysctl
        shell: sysctl -p
```

（4）基于列表 items 多个值创建用户，通过{{}}定义列表变量，with_items 选项传入变

```
[root@localhost ~]# ansible-playbook sysctl.yaml

PLAY [all] ************************************************************

TASK [Linux kernel config 2017] ***************************************
changed: [192.168.149.129]
changed: [192.168.149.128]

RUNNING HANDLER [source sysctl] ***************************************
changed: [192.168.149.129]
changed: [192.168.149.128]

PLAY RECAP ************************************************************
192.168.149.128            : ok=2    changed=2    unreachable=0
192.168.149.129            : ok=2    changed=2    unreachable=0
```

图 21-35　Ansible PlayBook 内核参数优化

量的值,代码如下,执行结果如图 21-36 所示。

```
- hosts: all
  remote_user: root
  tasks:
  - name: Linux system Add User list.
    user: name = {{ item }} state = present
    with_items:
       - jfedu1
       - jfedu2
       - jfedu3
       - jfedu4
```

```
[root@localhost ~]# ansible-playbook user.yaml

PLAY [all] ************************************************************

TASK [Linux system Add User list.] ************************************
changed: [192.168.149.129] => (item=jfedu1)
changed: [192.168.149.129] => (item=jfedu2)
changed: [192.168.149.129] => (item=jfedu3)
changed: [192.168.149.129] => (item=jfedu4)

PLAY RECAP ************************************************************
192.168.149.129            : ok=1    changed=1    unreachable=0

[root@localhost ~]#
```

(a) Ansible PlayBook item变量创建用户(1)

```
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/no
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
apache:x:48:48:Apache:/var/www:/sbin/nologin
exim:x:93:93::/var/spool/exim:/sbin/nologin
sdfjsdklfskl
jfedu001:x:501:501::/home/jfedu001:/bin/bash
redis:x:496:496:Redis Server:/var/lib/redis:/sbin/nologin
a:x:502:502::/home/a:/bin/bash
zabbix:x:503:503::/home/zabbix:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
jfedu1:x:504:504::/home/jfedu1:/bin/bash
jfedu2:x:505:505::/home/jfedu2:/bin/bash
jfedu3:x:506:506::/home/jfedu3:/bin/bash
jfedu4:x:507:507::/home/jfedu4:/bin/bash
```

(b) Ansible PlayBook item变量创建用户(2)

图 21-36　Ansible PlayBook item 变量创建用户

（5）Ansible PlayBook 可以自定义 template 模板文件，模板文件主要用于服务器需求不一致、需要独立定义的情况。例如两台服务器安装了 Nginx，安装完毕之后将服务器 A 的 HTTP 端口改成 80，服务器 B 的 HTTP 端口改成 81，基于 tempalte 模块可以轻松实现，方法步骤如下。

- Ansible hosts 文件指定不同服务器不同 httpd_port 端口，代码如下：

```
[web]
192.168.149.128 httpd_port = 80
192.168.149.129 httpd_port = 81
```

- Ansible 创建 nginx. conf jinja2 模板文件，cp nginx. conf nginx. conf. j2，并修改 listen 80 为 listen {{httpd_port}}，Nginx 其他配置项不变，代码如下：

```
cp nginx.conf nginx.conf.j2
listen {{httpd_port}};
```

- Ansible PlayBook 剧本 yaml 文件创建，代码如下：

```
- hosts: all
  remote_user: root
  tasks:
        - name: Nginx server Install 2017
          file: path = /usr/local/nginx/ state = directory
          notify:
                - nginx install
                - nginx config
  handlers:
        - name: nginx install
          shell: cd /tmp; rm - rf nginx - 1.12.0.tar.gz; wget http://nginx.org/download/nginx -
1.12.0.tar.gz; tar xzf nginx - 1.12.0
.tar.gz; cd nginx - 1.12.0; ./configure -- prefix = /usr/local/nginx; make; make install
        - name: nginx config
          template: src = /data/sh/nginx.conf.j2 dest = /usr/local/nginx/conf/nginx.conf
```

- Ansible PlayBook 执行剧本文件，结果如图 21-37 所示。

```
[root@localhost ~]# ansible-playbook nginx_template.yaml

PLAY [all] ****************************************************************

TASK [Nginx server Install 2017] ******************************************
changed: [192.168.149.129]
changed: [192.168.149.128]

RUNNING HANDLER [nginx install] *******************************************
changed: [192.168.149.129]
changed: [192.168.149.128]

RUNNING HANDLER [nginx config] ********************************************
changed: [192.168.149.129]
changed: [192.168.149.128]
```

(a) Ansible PlayBook执行模板yaml

图 21-37　Ansible PlayBook 执行剧本文件

```
keepalive_timeout  65;

#gzip  on;

server {
    listen  80;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        root   html;
        index  index.html index.htm;
```

(b) 149.128服务器Nginx HTTP port 80

```
keepalive_timeout  65;

#gzip  on;

server {
    listen  81;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        root   html;
        index  index.html index.htm;
    }
```

(c) 149.129服务器Nginx HTTP port 81

图 21-37 （续）

21.16 Ansible 配置文件详解

Ansible 默认配置文件为/etc/ansible/ansible.cfg，配置文件中可以对 Ansible 进行各项参数的调整，包括并发线程、用户、模块路径、配置优化等，ansible.cfg 常用参数详解如下：

- [defaults]：通用默认配置段。
- inventory = /etc/ansible/hosts：被控端 IP 或者 DNS 列表。
- library = /usr/share/my_modules/：Ansible 默认搜寻模块的位置。
- remote_tmp = $HOME/.ansible/tmp：Ansible 远程执行临时文件。
- pattern = *：对所有主机通信。
- forks = 5：并行线程数。
- poll_interval = 15：回频率或轮训间隔时间。
- sudo_user = root：sudo 远程执行用户名。
- ask_sudo_pass = True：使用 sudo，是否需要输入密码。
- ask_pass = True：是否需要输入密码。
- transport = smart：通信机制。
- remote_port = 22：远程 SSH 端口。
- module_lang = C：模块和系统之间通信的语言。
- gathering = implicit：控制默认 facts 收集（远程系统变量）。

- roles_path= /etc/ansible/roles：用于 PlayBook 搜索 Ansible roles。
- host_key_checking = False：检查远程主机密钥。
- #sudo_exe = sudo：sudo 远程执行命令。
- #sudo_flags = -H：传递 sudo 之外的参数。
- timeout = 10：SSH 超时时间。
- remote_user = root：远程登录用户名。
- log_path = /var/log/ansible.log：日志文件存放路径。
- module_name = command：Ansible 命令执行默认的模块。
- #executable = /bin/sh：执行的 shell 环境，用户 shell 模块。
- #hash_behaviour = replace：特定的优先级覆盖变量。
- #jinja2_extensions：允许开启 jinja2 拓展模块。
- #private_key_file = /path/to/file：私钥文件存储位置。
- #display_skipped_hosts = True：显示任何跳过任务的状态。
- #system_warnings = True：禁用系统运行 Ansible 潜在问题警告。
- #deprecation_warnings = True：PlayBook 输出禁用“不建议使用”警告。
- #command_warnings = False：command 模块 Ansible 默认发出警告。
- #nocolor = 1：输出带上颜色区别，0 表示开启，1 表示关闭。
- pipelining = False：开启 pipe SSH 通道优化。
- [accelerate]：accelerate 缓存加速。
- accelerate_port = 5099：加速连接端口 5099。

  accelerate_timeout = 30：命令执行超时时间，单位为 s。

  accelerate_connect_timeout = 5.0：上一个活动连接的时间，单位为 min。

  accelerate_daemon_timeout = 30：允许多个私钥被加载到 daemon。

  accelerate_multi_key = yes：任何客户端要想连接 daemon 都需要开启这个选项。

21.17 Ansible 性能调优

Ansible 企业生产环境中，如果管理的服务器越来越多，Ansibe 执行效率会变得比较慢，可以通过优化 Ansible 提高工作效率，由于 Ansible 基于 SSH 协议通信，SSH 连接慢会导致整个基于 Ansible 执行变得缓慢，也需要对 OpenSSH 进行优化，具体优化的方法如下。

(1) Ansible SSH 关闭秘钥检测。

默认以 SSH 登录远程客户端服务器，会检查远程主机的公钥(public key)，并将该主机的公钥记录在～/.ssh/known_hosts 文件中。下次访问相同主机时，OpenSSH 会核对公钥，如果公钥不同，OpenSSH 会发出警告，如果公钥相同，则提示输入密码。

SSH 对主机的 public_key 的检查等级是根据 StrictHostKeyChecking 变量来设定的，StrictHostKeyChecking 检查级别包括 no(不检查)、ask(询问)、yes(每次都检查)、False(关

闭检查）。

Ansible 配置文件中加入如下代码，即可关闭 StrictHostKeyChecking 检查：

```
host_key_checking = False
```

（2）OpenSSH 连接优化。

使用 OpenSSH 服务时，默认服务器端配置文件 UseDNS=YES 状态，该选项会导致服务器根据客户端的 IP 地址进行 DNS PTR 反向解析，得到客户端的主机名，然后根据获取到的主机名进行 DNS 正向 A 记录查询，并验证该 IP 是否与原始 IP 一致。关闭 DNS 解析代码如下：

```
sed -i '/^GSSAPI/s/yes/no/g; /UseDNS/d; /Protocol/aUseDNS no' /etc/ssh/sshd_config
/etc/init.d/sshd restart
```

（3）SSH pipelining 加速 Ansible。

SSH pipelining 是一个加速 Ansible 执行速度的简单方法，SSH pipelining 默认是关闭的，关闭是为了兼容不同的 sudo 配置，主要是 requiretty 选项。

如果不使用 sudo 建议开启该选项，打开此选项可以减少 Ansible 执行没有文件传输时，SSH 在被控机器上执行任务的连接数。使用 sudo 操作的时候，必须在所有被管理的主机上将配置文件/etc/sudoers 中 requiretty 选项禁用，代码如下：

```
sed    -i    '/^pipelining/s/False/True/g'    /etc/ansible/ansible.cfg
```

（4）Ansible facts 缓存优化。

Ansible PlayBook 在执行过程中，默认会执行 Gather facts，如果不需要获取客户端的 fact 数据的话，可以关闭获取 fact 数据功能，关闭之后可以加快 Ansible PlayBook 的执行效率。如需关闭 fact 功能，在 PlayBook YAML 文件中加入如下代码即可：

```
gather_facts: nogather_facts: no
```

Ansible facts 组件主要用于收集客户端设备的基础静态信息，这些信息可以在做配置管理的时候方便引用。facts 信息直接当作 Ansible PlayBook 变量信息进行引用，通过定制 facts 以便收集到想要的信息，同时可以通过 facter 和 ohai 来拓展 facts 信息，也可以将 facts 信息存入 Redis 缓存中，以下为 facts 使用 Redis 缓存的步骤。

（1）部署 Redis 服务，代码如下：

```
wget    http://download.redis.io/releases/redis-2.8.13.tar.gz
tar     zxf          redis-2.8.13.tar.gz
cd      redis-2.8.13
make    PREFIX=/usr/local/redis install
cp      redis.conf    /usr/local/redis/
```

将/usr/local/redis/bin/目录加入至环境变量配置文件/etc/profile 末尾，然后 shell 终端执行 source /etc/profile 让环境变量生效，代码如下：

```
export PATH = /usr/local/redis/bin: $PATH
```

启动及停止 Redis 服务命令，代码如下：

```
nohup /usr/local/redis/bin/redis - server  /usr/local/redis/redis.conf &
```

（2）安装 Python Redis 模块，代码如下：

```
easy_install pip
pip install redis
```

（3）Ansible 整合 Redis 配置。

在配置文件/etc/ansible/ansible.cfg 的 defaluts 段中加入以下代码，如果 Redis 密码为 admin，则开启 admin 密码行，代码如下：

```
gathering = smart
fact_caching = redis
fact_caching_timeout = 86400
fact_caching_connection = localhost:6379
#fact_caching_connection = localhost:6379:0:admin
```

（4）测试 Redis 缓存。

Ansible PlayBook 执行 nginx_wget.yaml 剧本文件，代码如下，结果如图 21-38 所示。

```
ansible - playbook    nginx_wget.yaml
```

```
[root@localhost ~]# ansible-playbook nginx_wget.yaml

PLAY [192.168.*] ****************************************

TASK [Gathering Facts] **********************************
ok: [192.168.149.129]
ok: [192.168.149.130]
ok: [192.168.149.131]
ok: [192.168.149.128]

TASK [www.jfedu.net Centos Manager] *********************
```

图 21-38 Ansible PlayBook 执行.yaml 剧本文件

检查 Redis 服务器，facts key 已存入 Redis 中，如图 21-39 所示。

```
[root@localhost ~]# redis-cli
127.0.0.1:6379>
127.0.0.1:6379> KEYS *
(empty list or set)
127.0.0.1:6379> KEYS *
1) "ansible_facts192.168.149.131"
2) "ansible_facts192.168.149.130"
3) "ansible_facts192.168.149.128"
4) "ansible_facts192.168.149.129"
5) "ansible_cache_keys"
127.0.0.1:6379>
```

图 21-39 Redis 缓存服务器缓存 facts 主机信息

（5）ControlPersist SSH 优化。

ControlPersist 特性需要高版本的 SSH 支持，CentOS 6 默认是不支持的，如果需要使用，需要自行升级 OpenSSH。

ControlPersist 即持久化的 socket，一次验证多次通信。并且只需要修改 SSH 客户端配置，也即 Ansible 被管理主机。可使用 YUM 或者源码编译升级 OpenSSH 服务，升级完毕 ControlPersist 的设置办法如下，在其用户的家目录创建 config 文件，如果 Ansible 以 root 用户登录客户端，只需要在客户端的/root/.ssh/config 目录中添加如下代码即可：

```
Host *
  Compression yes
  ServerAliveInterval 60
  ServerAliveCountMax 5
  ControlMaster auto
  ControlPath ~/.ssh/sockets/%r@%h-%p
  ControlPersist 4h
```

开启 ControlPersist 特性后，SSH 在建立 sockets 后，节省了每次验证和创建的时间，对 Ansible 执行速度提升是非常明显的。

第 22 章



Jenkins 持续集成企业实战

构建企业自动化部署平台，可以大大地提升企业网站部署效率，企业生产环境每天需要更新各种系统，传统更新网站的方法是使用 shell＋rsync 实现网站代码备份、更新，更新完之后，运维人员手动发送邮件给测试人员、开发人员以及相关的业务人员，传统更新网站耗费大量的人力，同时偶尔由于误操作会出现问题。构建自动化部署平台变得迫在眉睫。

本章向读者介绍传统网站部署方法、企业主流部署方法、Jenkins 持续集成简介、持续集成平台构建、Jenkins 插件部署、Jenkins 自动化部署网站、Jenkins 多实例及 Ansible＋Jenkins 批量自动部署等内容。

22.1　传统网站部署的流程

服务器网站部署是运维工程师的主要工作之一，传统运维网站部署主要靠手动部署，手工部署网站的流程大致分为需求分析→原型设计→开发代码→提交测试→内网部署→确认上线→备份数据→外网更新→发布完毕→网站测试等，如果发现外网部署的代码有异常，需要及时回滚，如图 22-1 所示。

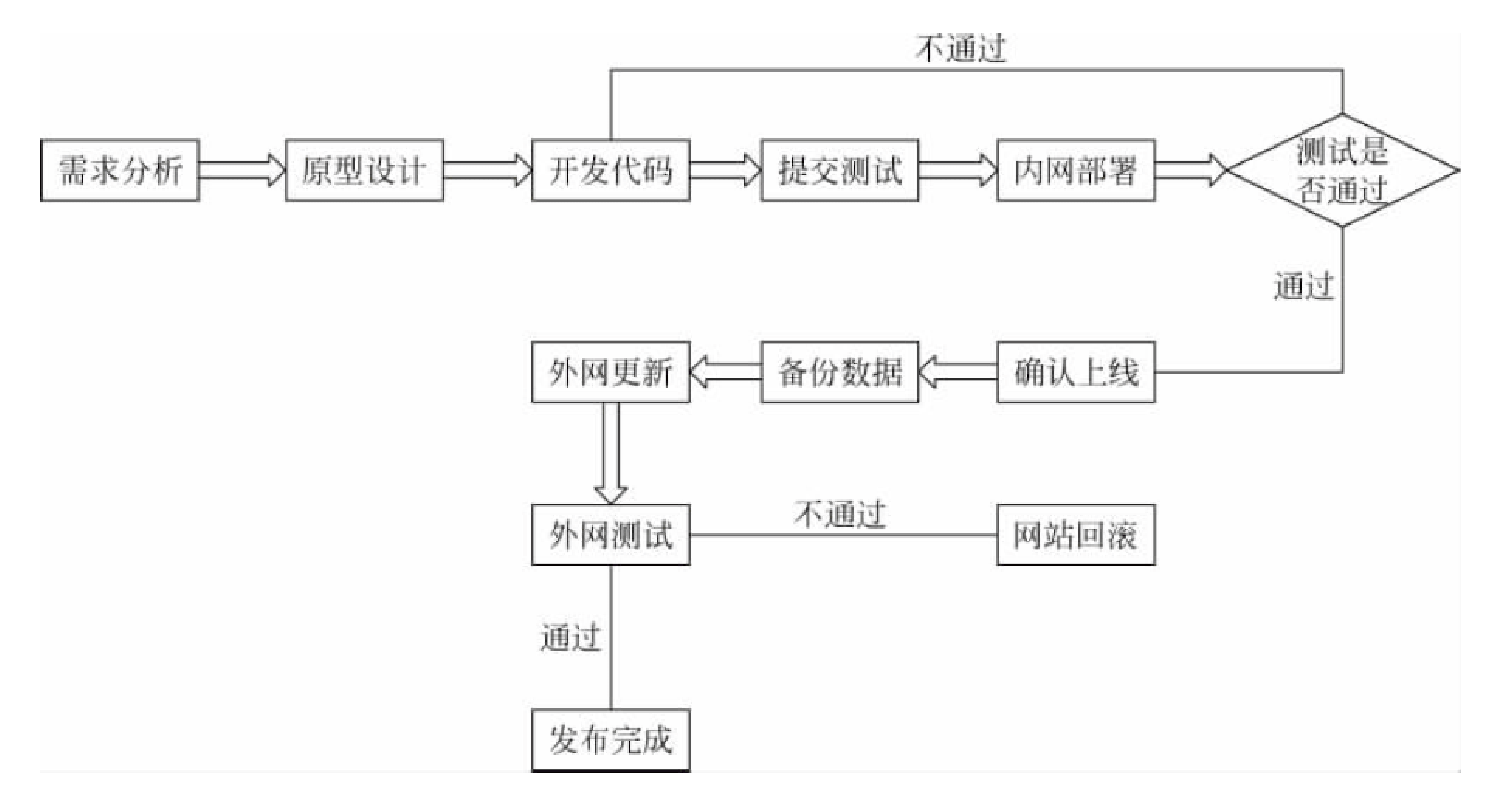


图 22-1　网站传统部署方法及流程

服务器部署基于 YUM 安装 LAMP 架构，并且部署 Discuz，最终效果如图 22-2 所示。

图 22-2 YUM 部署 LAMP+Discuz 网站

通过 SecureCRT 登录网站服务器，并将 logo. png 文件上传至网站目录，手动备份网站，并且更新网站的 logo，更新完毕如图 22-3 所示。



图 22-3 手工更新 LAMP 网站 logo 文件

22.2 目前主流网站部署的流程

传统部署网站的方法对于单台或者几台服务器更新很容易，如果服务器规模超过百台或者千台，更新网站代码很频繁，手工更新非常消耗人力和时间成本。

基于主流的 Hudson/Jenkins 工具平台实现全自动网站部署、网站测试、网站回滚会大大地减少网站部署的成本，Jenkins 的前身为 Hudson，Hduson 为商业版，Jenkins 为开源免费版。

Jenkins 是一个可扩展的持续集成引擎，是一个开源软件项目，旨在提供一个开放易用的软件平台，使软件的持续集成变成可能。而且 Jenkins 平台的安装和配置非常容易，使用也非常简单。构建 Jenkins 平台可以解放人员的双手，具体内容如下：

- 开发人员：对于开发人员来说，只需负责网站代码的编写，不需要手动再对源码进行编译、打包、单元测试等工作，开发人员直接将写好的代码分支存放在 SVN、GIT 仓库即可。
- 运维人员：对于运维人员来说，使用 Jenkins 自动部署，可以减轻人工干预的错误率，同时解放运维人员繁杂的上传代码、手动备份、手动更新。
- 测试人员：对于测试人员来说，可以通过 Jenkins 进行代码测试、网站功能或者性能测试。

基于 Jenkins 自动部署网站的流程大致分为需求分析→原型设计→开发代码→提交测试→Jenkins 内网部署→确认上线→Jenkins 备份数据→Jenkins 外网部署→发布完毕→Jenkins 网站测试等，如果发现外网部署的代码有异常，可以通过 Jenkins 及时回滚，如图 22-4 所示。

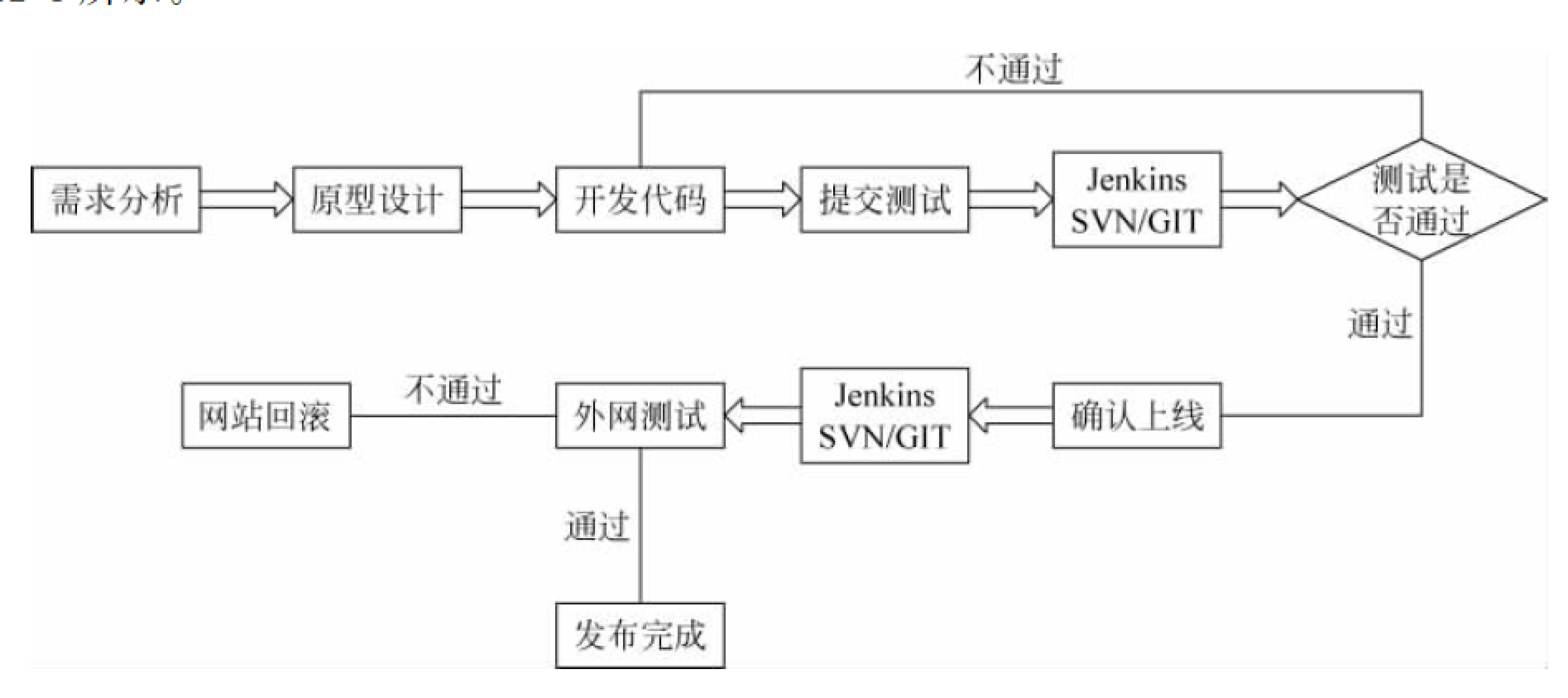


图 22-4　Jenkins 部署网站的方法及流程

22.3　Jenkins 持续集成简介

持续集成(continuous integration，CI)是一种软件开发实践，对于提高软件开发效率并保障软件开发质量提供了理论基础，持续集成意义如下：

- 持续集成中的任何一个环节都是自动完成的，无须太多的人工干预，有利于减少重复过程以节省时间、费用和工作量；
- 持续集成保障了每个时间点上团队成员提交的代码是能成功集成的，换言之，任何时间点都能第一时间发现软件的集成问题，使任意时间发布可部署的软件成为可能；

- 持续集成还能利于软件本身的发展趋势，在需求不明确或是频繁性变更的情景中尤其重要，持续集成的质量能帮助团队进行有效决策，同时建立团队对开发产品的信心。

22.4 Jenkins 持续集成组件

要掌握 Jenkins 技能，需要了解 Jenkins 持续集成平台依赖的组件，例如 JOB 工程、SVN 仓库源、Jenkins 服务器，详解如下：

- 自动构建过程 JOB，JOB 的功能主要是获取 SVN/GIT 源码、自动编译、自动打包、部署分发和自动测试等。
- 源代码存储库，开发编写代码需上传至 SVN、GIT 代码库中，供 Jenkins 来获取。
- Jenkins 持续集成服务器，用于部署 Jenkins UI、存放 JOB 工程、各种插件、编译打包的数据等。

22.5 Jenkins 平台安装部署

Jenkins 官网免费获取 Jenkins 软件，官网地址 http://mirrors.jenkins-ci.org/下载稳定的 Jenkins 版本。Jenkins 是基于 Java 开发的一种持续集成工具，所以 Jenkins 服务器需安装 Java JDK 开发软件。Jenkins 平台搭建步骤如下。

(1) Jenkins 稳定版下载，地址如下。

http://updates.jenkins-ci.org/download/war/1.651.2/jenkins.war

(2) 官网下载 Java JDK，并解压安装，代码如下：

```
tar -xzf jdk-7u25-linux-x64.tar.gz ; mkdir -p /usr/java/ ; mv jdk1.7.0_25/ /usr/java/
```

(3) 配置 Java 环境变量，在/etc/profile 配置文件中末尾加入如下代码：

```
export JAVA_HOME=/usr/java/jdk1.7.0_25
export CLASSPATH=$CLASSPATH:$JAVA_HOME/lib:$JAVA_HOME/jre/lib
export PATH=$JAVA_HOME/bin:$JAVA_HOME/jre/bin:$PATH:$HOMR/bin
```

执行如下代码使其环境变量生效，并查看环境变量，命令如下：

```
source /etc/profile
java --version
```

(4) Tomcat Java 容器配置，代码如下：

```
wget http://mirror.bit.edu.cn/apache/tomcat/tomcat-6/v6.0.53/bin/apache-tomcat-6.0.53.
tar.gz
tar xzf apache-tomcat-6.0.53.tar.gz
mv apache-tomcat-6.0.53   /usr/local/tomcat
```

（5）Tomcat 发布 Jenkins，将 jenkins.war 复制到 Tomcat 默认发布的目录下，并使用 jar 工具解压，启动 Tomcat 服务即可，代码如下：

```
rm     -rf /usr/local/tomcat/webapps/*
mkdir  -p /usr/local/tomcat/webapps/ROOT/
mv     jenkins.war /usr/local/tomcat/webapps/ROOT/
cd     /usr/local/tomcat/webapps/ROOT/
jar    -xvf jenkins.war; rm -rf Jenkins.war
sh     /usr/local/tomcat/bin/startup.sh
```

（6）通过客户端浏览器访问 Jenkins 服务器 IP 地址，如图 22-5 所示。



图 22-5　Jenkins 自动部署平台

22.6　Jenkins 相关概念

要熟练掌握 Jenkins 持续集成的配置、使用和管理，需要了解相关的概念，例如代码开发、编译、打包、构建等名称概念，常见的代码相关概念包括 make、ant、maven、Eclipse、Jenkins 等。具体内容如下：

（1）make 编译工具。

make 编译工具是 Linux 和 Windows 最原始的编译工具，在 Linux 下编译程序常用 make，Windows 下对应的工具为 nmake。读取本地 makefile 文件，该文件决定了源文件之间的依赖关系，make 负责根据 makefile 文件去组织构建软件，负责指挥编译器如何编译，连接器如何连接，以及最后生成可用二进制的代码。

（2）ant 编译工具。

make 工具在编译比较复杂的工程中时使用起来不方便，语法很难理解，因此延伸出 ant 工具。ant 工具属于 Apache 基金会软件成员之一，是一个将软件编译、测试、部署等步骤联系在一起加以自动化的一个工具，大多用于 Java 环境中的软件开发。

ant 构建文件是 XML 文件。每个构建文件定义一个唯一的项目(project 元素)。每个项目下可以定义很多目标元素,这些目标之间可以有依赖关系。

构建一个新的项目时,应该编写 ant 构建文件。因为构建文件定义了构建过程,并为团队开发中每个人所使用。

ant 构建文件默认名为 build. xml,也可以修改为其他的名称。只不过在运行的时候需把这个命名当作参数传给 ant。构建文件可以放在任何的位置,一般做法是放在项目顶层目录也即根目录,这样可以保持项目的简洁和清晰。

(3) maven 编译工具。

maven 工具是对 ant 工具的进一步改进,在 make 工具中,如果需要编译某些源文件,首先要安装编译器等工具。有时候需要不同版本的编译器,例如 Java 编译器在编译文件时需要各种依赖包的支持,如果把每个包都下载下来,在 makefile 中进行配置制定,当需要的包非常多时,很难管理。

maven 与 ant 类似,也是构建(build)工具,它是如何调用各种不同的编译器连接器呢?使用 maven plugin(maven 插件),maven 项目对象模型 POM(project object model),可以通过一小段描述信息来管理项目的构建,报告和文档的软件项目管理工具。maven 除了以程序构建能力为特色之外,还提供高级项目管理工具。

POM 是 maven 项目中的文件,使用 XML 表示,名称为 pom. xml。在 maven 中,当构建 project 的时候,不仅仅是一堆包含代码的文件,还包含 pom. xml 配置文件,该文件包括 project 与开发者有关的、缺陷跟踪系统、组织与许可、项目的 URL、项目依赖以及其他配置。

基于 maven 构建编译时,project 可以什么都没有,甚至没有代码,但是必须包含 pom. xml 文件。由于 maven 的默认构建规则有较高的可重用性,所以常常用两三行 maven 构建脚本就可以构建简单的项目。

由于 maven 的面向项目的方法,许多 Apache Jakarta 项目发文时使用 maven,而且公司项目采用 maven 的比例在持续增长。

(4) Jenkins 框架工具。

maven 可以实现对软件代码进行编译、打包、测试,功能已经很强大了,那还需要 Jenkins 做什么呢? maven 可以控制编译,控制连接,可以生成各种报告,可以进行代码测试。但是默认不能控制完整的流程。没有顺序定义,那是先编译还是先连接,先进行代码测试还是先生成报告? 因此需要使用脚本来对 maven 进行控制,实现这些流程的控制。

(5) Eclipse 工具。

Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台。就其本身而言,它只是一个框架和一组服务,用于通过插件组件构建开发环境。Eclipse 附带了一个标准的插件集,包括 Java 开发工具(Java development kit,JDK),主要用于开发者开发网站代码。

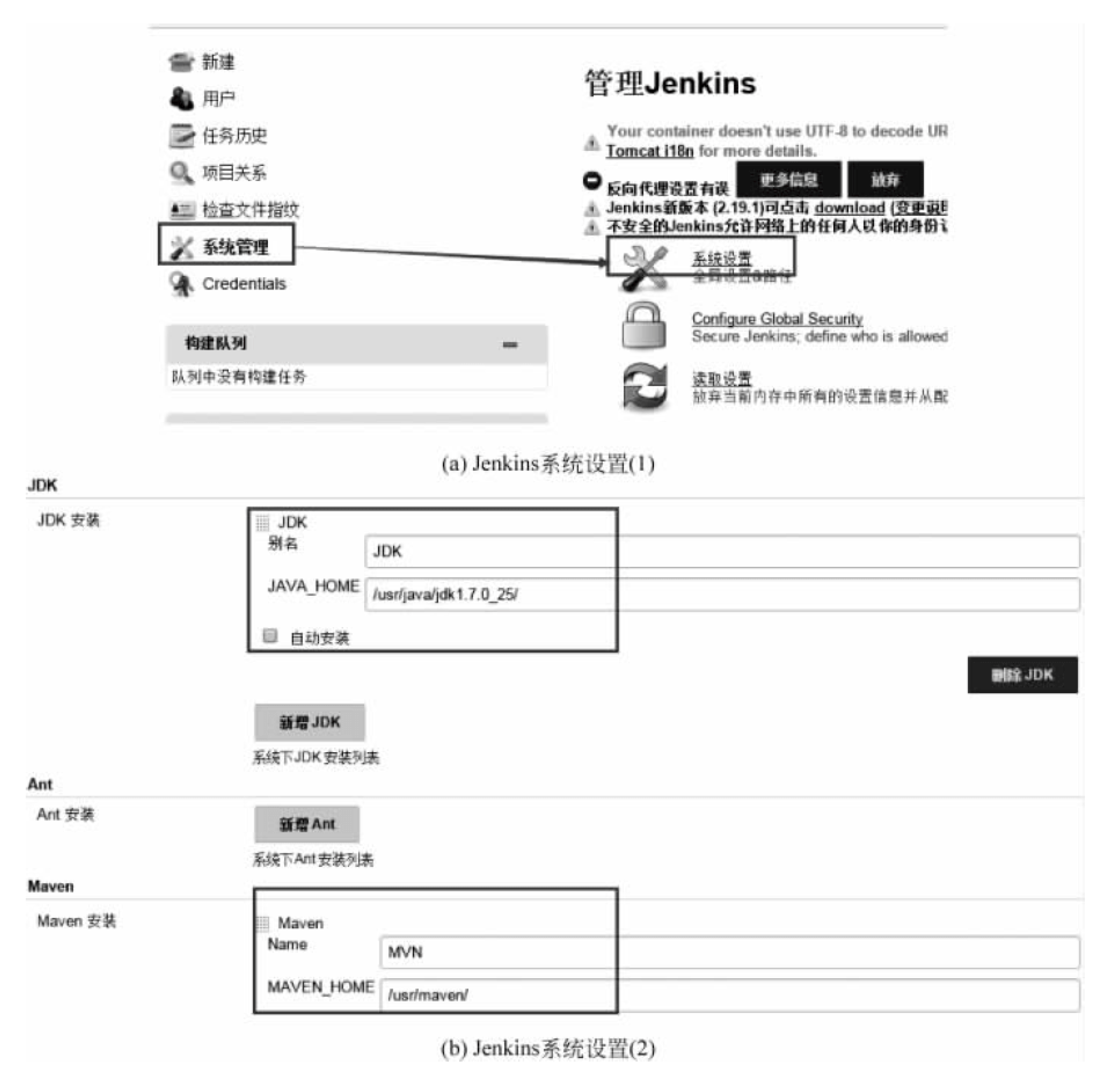
22.7 Jenkins 平台设置

Jenkins 持续基础平台部署完毕，需要进行简单配置，例如配置 Java 路径、安装 maven、指定 SVN、GIT 仓库地址等，以下为 Java 路径和 maven 设置步骤。

（1）Jenkins 服务器安装 maven，代码如下：

```
wget
http://mirrors.tuna.tsinghua.edu.cn/apache/maven/maven-3/3.3.9/binaries/apache-maven-
3.3.9-bin.tar.gz
tar -xzf apache-maven-3.3.9-bin.tar.gz
mv apache-maven-3.3.9   /usr/maven/
```

（2）Jenkins 系统设置环境变量，如图 22-6 所示。



(a) Jenkins系统设置(1)

(b) Jenkins系统设置(2)

图 22-6　Jenkins 系统设置

(3) Jenkins 系统设置完毕,需创建 JOB 工程,具体步骤如下:

在 Jenkins 平台首页中创建一个新任务,填入 Item 名称,选择“构建一个 maven 项目”,再单击 OK 按钮,如图 22-7 所示。



图 22-7 Jenkins 创建 Jenkins JOB 工程

(4) 创建完 JOB 任务,需对任务进行配置,如图 22-8 所示。

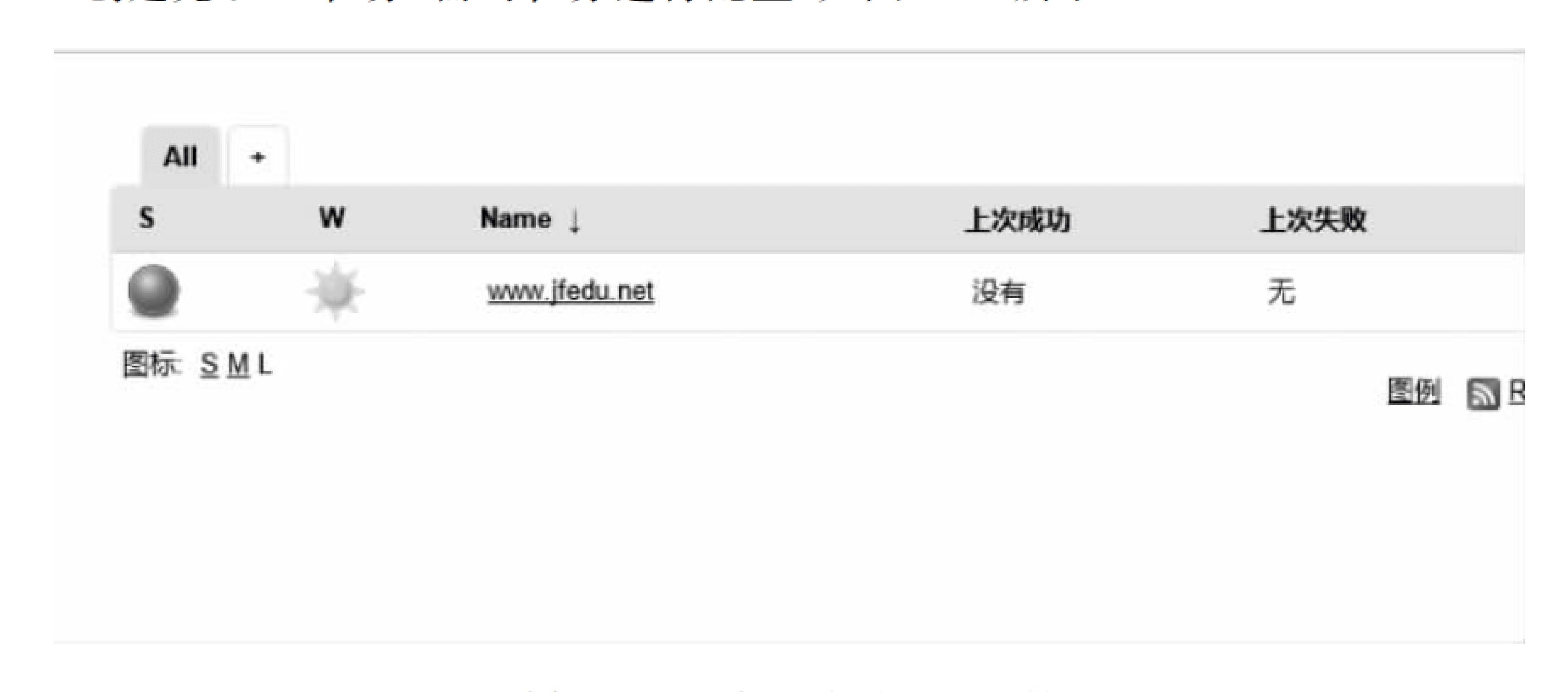


图 22-8 Jenkins 配置 JOB 工程

(5) 单击 www.jfedu.net 工程名,选择“配置→JOB 工程详细配置→源码管理→Subversion”配置 SVN 仓库地址,如果报错需要输入 SVN 用户名和密码,如图 22-9 所示。

源码管理,SVN 代码迁出参数详解如下:

- Repository url:配置 SVN 仓库地址。
- Local module directory:存储 SVN 源码的路径。
- Ignore externals option:忽略额外参数。
- Check-out Strategy:代码检出策略。
- Repository browser:仓库浏览器,默认 Auto。
- add more locations:源码管理,允许下载多个地址的代码。



图 22-9　Jenkins 配置 SVN 仓库地址

- Repository depth：获取 SVN 源码的目录深度，默认为 infinity。
- empty：不检出项目的任何文件。
- files：所有文件。
- immediates：目录第一级。
- infinity：整个目录所有文件。

（6）配置 maven 编译参数，依次选择 Build→Goals and options，并输入 clean install -Dmaven.test.skip=true，此处为 maven 自动编译、打包并跳过单元测试选项，如图 22-10 所示。

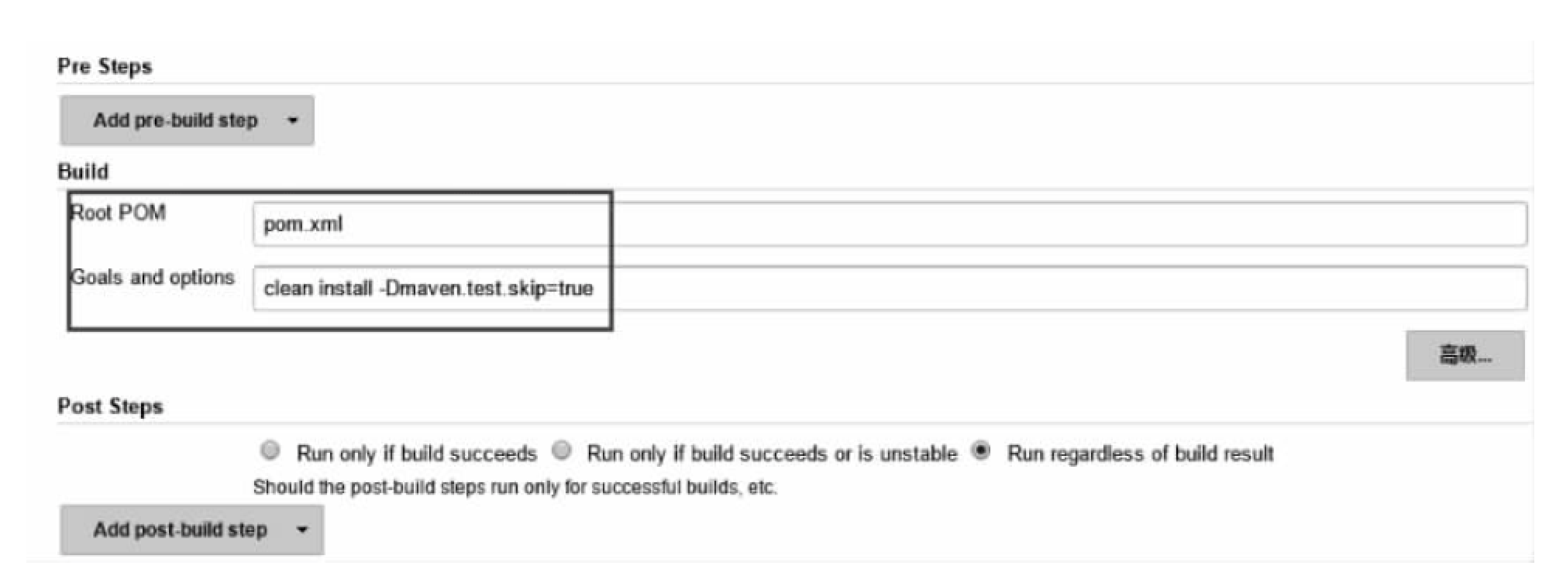


图 22-10　Jenkins 配置 maven 编译参数

maven 工具常用命令详解如下：

- mvn clean：打包清理（删除 target 目录内容）。
- mvn compile：编译项目。
- mvn package：打包发布。

- mvn package -Dmaven. test. skip=ture：打包时跳过测试。

通过以上的配置步骤，即完成了 JOB 工程的创建。

22.8 Jenkins 构建 JOB 工程

Jenkins JOB 工程创建完毕，直接运行构建，Jenkins 将从 SVN 仓库获取 Web 代码，然后通过 maven 编译、打包，并最终生成可以使用的 war 包即可，操作步骤如下。

(1) 单击 www.jfedu.net 工程名，进入 JOB 工程详细配置界面，单击"立即构建"，如图 22-11 所示。



图 22-11 Jenkins JOB 工程配置界面

(2) 查看 Build History，单击最新一次百分比滚动条任务，如图 22-12 所示。



图 22-12 Jenkins JOB 工程 Build 界面

(3) 进入 JOB 工程编译详细页面，单击 Console Output，如图 22-13 所示。

(4) 查看 Jenkins 构建实时日志，如图 22-14 所示。

控制台日志打印 Finished：SUCCESS，则表示 Jenkins 持续集成构建完成，会在 Jenkins 服务器目录 www.jfedu.net 工程名目录下生成网站可用的 war 文件，将该 war 包



图 22-13　Jenkins JOB 工程 Console Output 界面

```
Started by user anonymous
Building on master in workspace /root/.jenkins/workspace/www.jfedu.net
Updating svn://139.224.227.121:8801/edu at revision '2017-06-04T15:17:11.704 +0800'
At revision 200
no change for svn://139.224.227.121:8801/edu since the previous build
No emails were triggered.
Parsing POMs
[www.jfedu.net] $ /usr/java/jdk1.8.0_131//bin/java -cp /root/.jenkins/plugins/maven-plugin/WEB-INF/lib/maven31-ag
1.5.jar:/data/maven/boot/plexus-classworlds-2.5.2.jar:/data/maven/conf/logging jenkins.maven3.agent.Maven31Main /
/usr/local/tomcat_jenkins/webapps/ROOT/WEB-INF/lib/remoting-2.57.jar /root/.jenkins/plugins/maven-plugin/WEB-INF/
/root/.jenkins/plugins/maven-plugin/WEB-INF/lib/maven3-interceptor-commons-1.5.jar 27365
<===[JENKINS REMOTING CAPACITY]===>channel started
Executing Maven:  -B -f /root/.jenkins/workspace/www.jfedu.net/pom.xml clean install -Dmaven.test.skip=true
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building edu Maven Webapp 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ edu ---
[INFO] Deleting /root/.jenkins/workspace/www.jfedu.net/target
```

(a) Jenkins JOB工程编译控制台(1)

```
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ edu ---
[INFO] Installing /root/.jenkins/workspace/www.jfedu.net/target/edu.war to /root/.m2/repository/com/shareku/edu/0
SNAPSHOT.war
[INFO] Installing /root/.jenkins/workspace/www.jfedu.net/pom.xml to /root/.m2/repository/com/shareku/edu/0.0.1-SN
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 13.733 s
[INFO] Finished at: 2017-06-04T15:17:35+08:00
[INFO] Final Memory: 25M/171M
[INFO] ------------------------------------------------------------------------
[JENKINS] Archiving /root/.jenkins/workspace/www.jfedu.net/pom.xml to com.shareku/edu/0.0.1-SNAPSHOT/edu-0.0.1-SN
[JENKINS] Archiving /root/.jenkins/workspace/www.jfedu.net/target/edu.war to com.shareku/edu/0.0.1-SNAPSHOT/edu-0
channel stopped
[www.jfedu.net] $ /bin/sh -xe /usr/local/tomcat_jenkins/temp/hudson6629705887694115145.sh
Archiving artifacts
```

(b) Jenkins JOB工程编译控制台(2)

图 22-14　Jenkins JOB 工程编译控制台

部署至其他服务器即可，war 路径为/root/.jenkins/workspace/www.jfedu.net/target/edu.war。

至此，Jenkins 持续集成平台自动构建软件完成，该步骤只是生成了 war 包，并没有实现

自动将该 war 包部署至其他服务器，如果要自动部署需要基于 Jenkins 插件或者 shell、Python 等自动化部署脚本。

22.9　Jenkins 自动化部署

通过手动构建 Jenkins JOB 工程、自动编译、打包生成 war 包，并不能实现自动部署，要实现自动部署可以使用自动部署插件或者 shell 脚本、Python 脚本等。

以下为以 shell 脚本实现 Jenkins 自动部署 war 至其他多台服务器，并自动启动 Tomcat，实现最终 Web 浏览器访问。Jenkins 自动部署完整操作步骤如下：

(1) 单击 www.jfedu.net 工程名，选择"配置→构建后操作→Add post-build step→Archive the artifacts→用于存档的文件"，输入 ** /target/ *.war，该选项主要用于 Jenkins 编译后会将 war 包存档一份到 target 目录，该文件可以通过 Jenkins Tomcat 的 HTTP 端口访问，如图 22-15 所示。



(a) Jenkins JOB工程自动部署设置(1)

(b) Jenkins JOB工程自动部署设置(2)

图 22-15　Jenkins JOB 工程自动部署设置

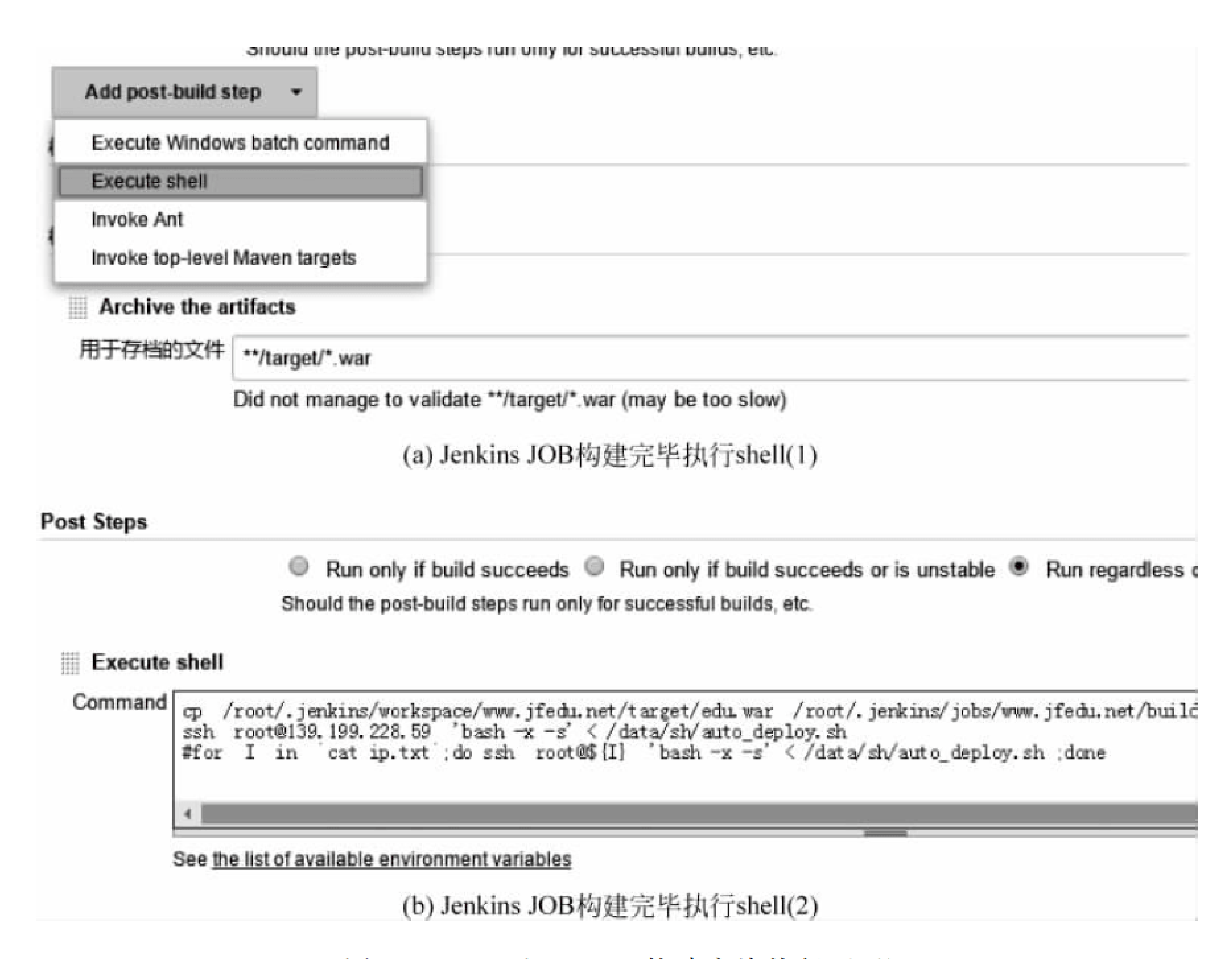
(2) Jenkins 构建完毕，访问 Jenkins war 存档的文件，URL 地址如下：

http://139.224.227.121:7001/job/www.jfedu.net/lastSuccessfulBuild/artifact/

target/edu. war

（3）依次 Add post-build step→Execute shell→Command，输入如下代码，实现 Jenkins edu. war 包自动部署，以下为 139. 199. 228. 59 客户端单台服务器部署 edu. war，如果多台可以使用 ip.txt 列表，将 IP 加入至 ip.txt，通过 for 循环实现批量部署，如图 22-16 所示。

```
cp /root/. jenkins/workspace/www. jfedu. net/target/edu. war /root/. jenkins/jobs/www. jfedu.
net/builds/lastSuccessfulBuild/archive/target/
ssh root@139.199.228.59 'bash - x - s' < /data/sh/auto_deploy. sh
#for I in 'cat ip. txt'; do ssh root@ ${I} 'bash - x - s' < /data/sh/auto_deploy. sh ; done
```



(a) Jenkins JOB构建完毕执行shell(1)

(b) Jenkins JOB构建完毕执行shell(2)

图 22-16　Jenkins JOB 构建完毕执行 shell

（4）Jenkins 将 edu. war 自动部署至 139. 199. 228. 59 服务器 Tomcat 发布目录，需提前配置登录远程客户端免秘钥，免秘钥配置首先在 Jenkins 服务器执行 ssh-keygen 命令，然后按 Enter 键生成公钥和私钥，再将公钥 id_rsa. pub 复制到客户端/root/. ssh/目录，并重命名为 authorized_keys，操作命令如下：

```
ssh - keygen - t rsa - P '' - f /root/. ssh/id_rsa
ssh - copy - id - i /root/. ssh/id_rsa. pub 139.199.228.59
```

（5）shell 脚本需放在 Jenkins 服务器/data/sh/，无须放在客户端，shell 脚本内容如下：

```
#!/bin/bash
#Auto deploy Tomcat for jenkins
#By author jfedu.net 2017
export JAVA_HOME=/usr/java/jdk1.6.0_25
TOMCAT_PID='/usr/sbin/lsof -n -P -t -i :8081'
TOMCAT_DIR="/usr/local/tomcat/"
FILES="edu.war"
DES_DIR="/usr/local/tomcat/webapps/ROOT/"
DES_URL="http://139.224.227.121:7001/job/www.jfedu.net/lastSuccessfulBuild/artifact/
target/"
BAK_DIR="/export/backup/'date +%Y%m%d-%H%M'"
[ -n "$TOMCAT_PID" ] && kill -9 $TOMCAT_PID
cd $DES_DIR
rm -rf $FILES
mkdir -p $BAK_DIR; \cp -a $DES_DIR/* $BAK_DIR/
rm -rf $DES_DIR/*
wget $DES_URL/$FILES
/usr/java/jdk1.6.0_25/bin/jar -xvf $FILES
####################
cd $TOMCAT_DIR; rm -rf work
/bin/sh $TOMCAT_DIR/bin/start.sh
sleep 10
tail -n 50 $TOMCAT_DIR/logs/catalina.out
```

如上通过 shell+for 循环可以实现网站简单部署，如果需要将 Jenkins edu.war 包批量快速部署至 100 台、500 台服务器，该如何去实现呢？后续小节会讲解到。

22.10 Jenkins 插件安装

Jenkins 最大的功能莫过于插件丰富，基于各种插件可以满足各项需求，Jenkins 本身是一个框架，真正发挥作用的是各种插件。Jenkins 默认自带很多插件，如果需添加新插件，可以在 Jenkins 平台主页面进行操作，操作步骤如下：

在 Jenkins 平台首页中选择“系统管理→管理插件→可选插件”，搜索 email-ext-plugin 插件选择并安装，如果没有该插件，则需单击“高级”按钮，手动上传插件并安装，操作步骤如图 22-17 所示。

访问 Jenkins 官网手动下载插件，将下载的插件上传至服务器 Jenkins 根目录(/root)下的 plugins 目录，即/root/.jenkins/plugins 目录，重启 Jenkins 即可。Jenkins 插件下载地址为 https://wiki.jenkins-ci.org/display/JENKINS/Plugins。安装 Email-ext-Plugin 邮件插件的方法如下。

(1) 下载 Email-ext、Token-macro、Email-template 插件，可以搜索某个插件，输入插件名称，如图 22-18 所示。

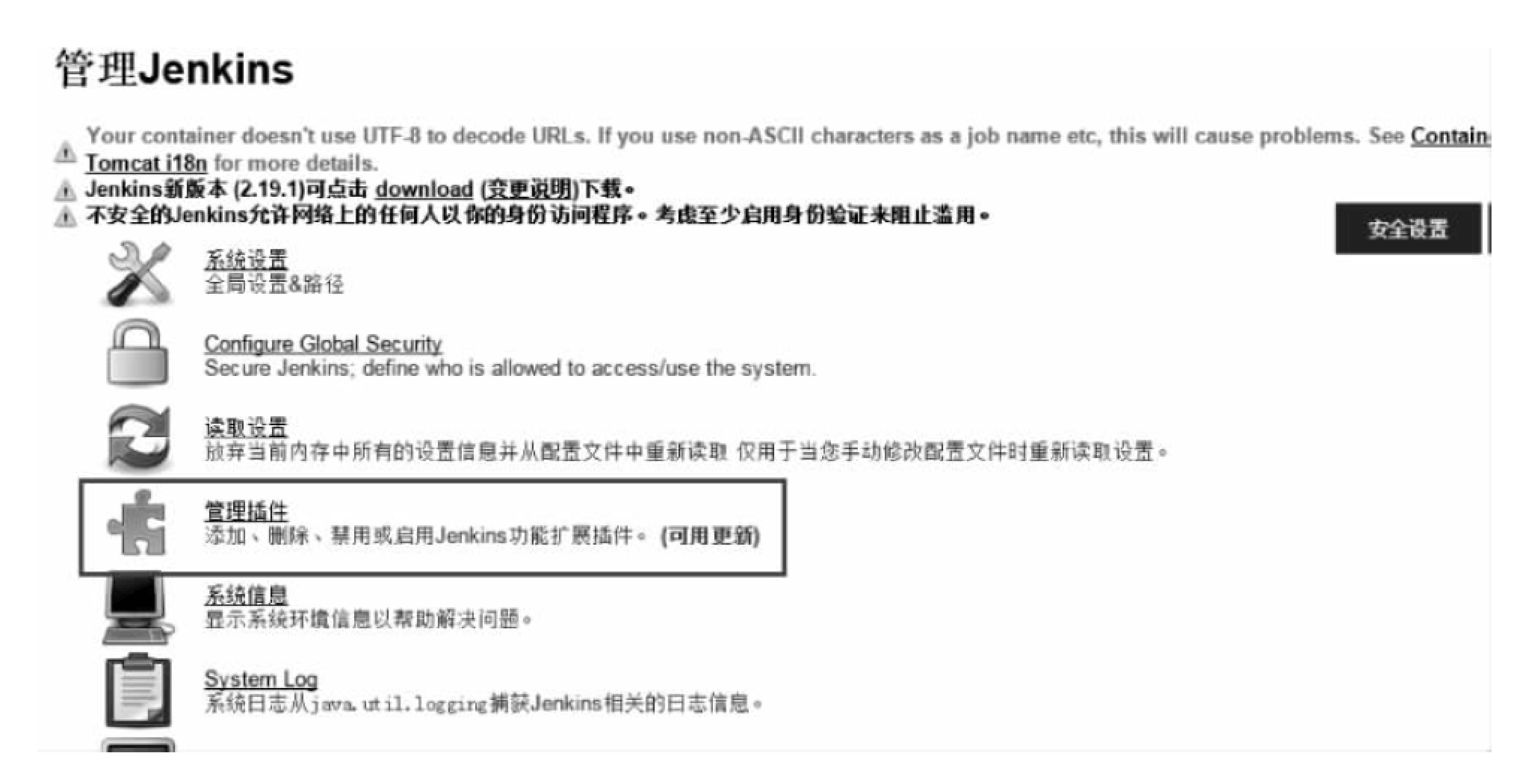


图 22-17 Jenkins 添加新插件

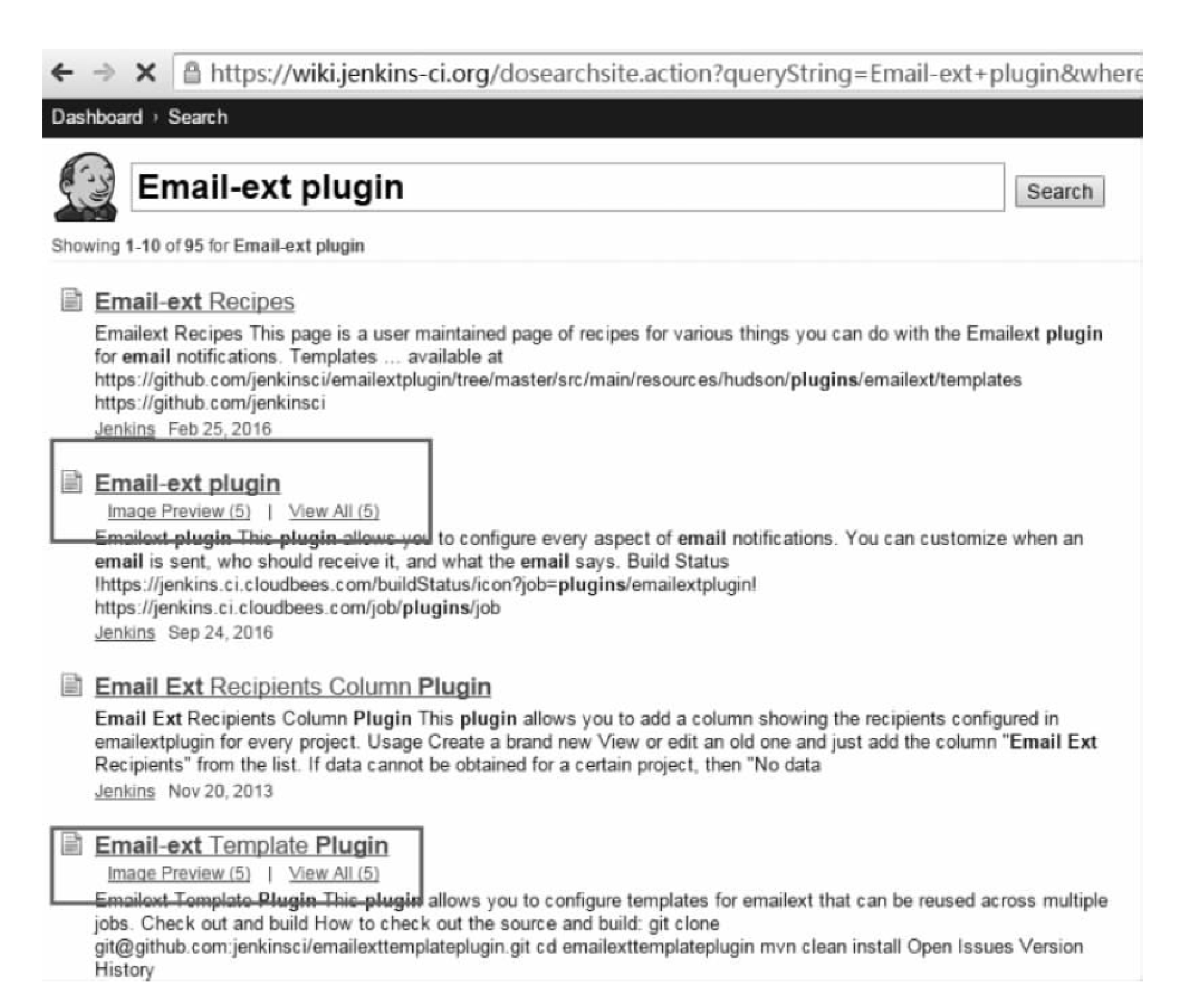


图 22-18 Jenkins 下载新插件

（2）Email-ext 和 Token-macro、Email-template 插件下载 URL 如下：

https://wiki.jenkins-ci.org/display/JENKINS/Email-ext+plugin

https://wiki.jenkins-ci.org/display/JENKINS/Token+Macro+Plugin

https://wiki.jenkins-ci.org/display/JENKINS/Email-ext+Template+Plugin

（3）安装 Token-macro 插件，如图 22-19 所示。



(a) Jenkins Token-macro插件安装(1)

(b) Jenkins Token-macro插件安装(2)

图 22-19　Jenkins Token-macro 插件安装

（4）安装 Email-ext 插件，如图 22-20 所示。

（5）Email-ext、Token-macro 和 Email-template 插件安装完毕，如图 22-21 所示。

（6）Email 插件安装完毕，在 Jenkins 主界面中选择“系统管理→系统设置”，若出现选项 Extended E-mail Notification，则表示 Jenkins Email 邮件插件安装完毕，如图 22-22 所示。

如需安装 GIT、Publish Over 插件或者安装 Jenkins 其他插件，与 Email 插件安装方法一致。



图 22-20　Jenkins Email-ext 插件安装

安装/更新 插件中

准备
token-macro 完成
准备
email-ext 完成
准备
emailext-template 完成

返回首页
(返回首页使用已经安装好的插件)

安装完成后重启Jenkins(空闲时)

图 22-21　Jenkins 插件安装完毕

Extended E-mail Notification
SMTP server
Default user E-mail suffix
Default Content Type　Plain Text (text/plain)
Use List-ID Email Header
Add 'Precedence: bulk' Email Header
Default Recipients
Reply To List
Emergency reroute

图 22-22　Jenkins Email 邮件插件

22.11　Jenkins 邮件配置

如上 Jenkins 持续集成平台配置完毕，可以进行网站代码的自动更新、部署、升级及回滚操作，通过控制台信息可以查看每个 JOB 工程构建的状态。

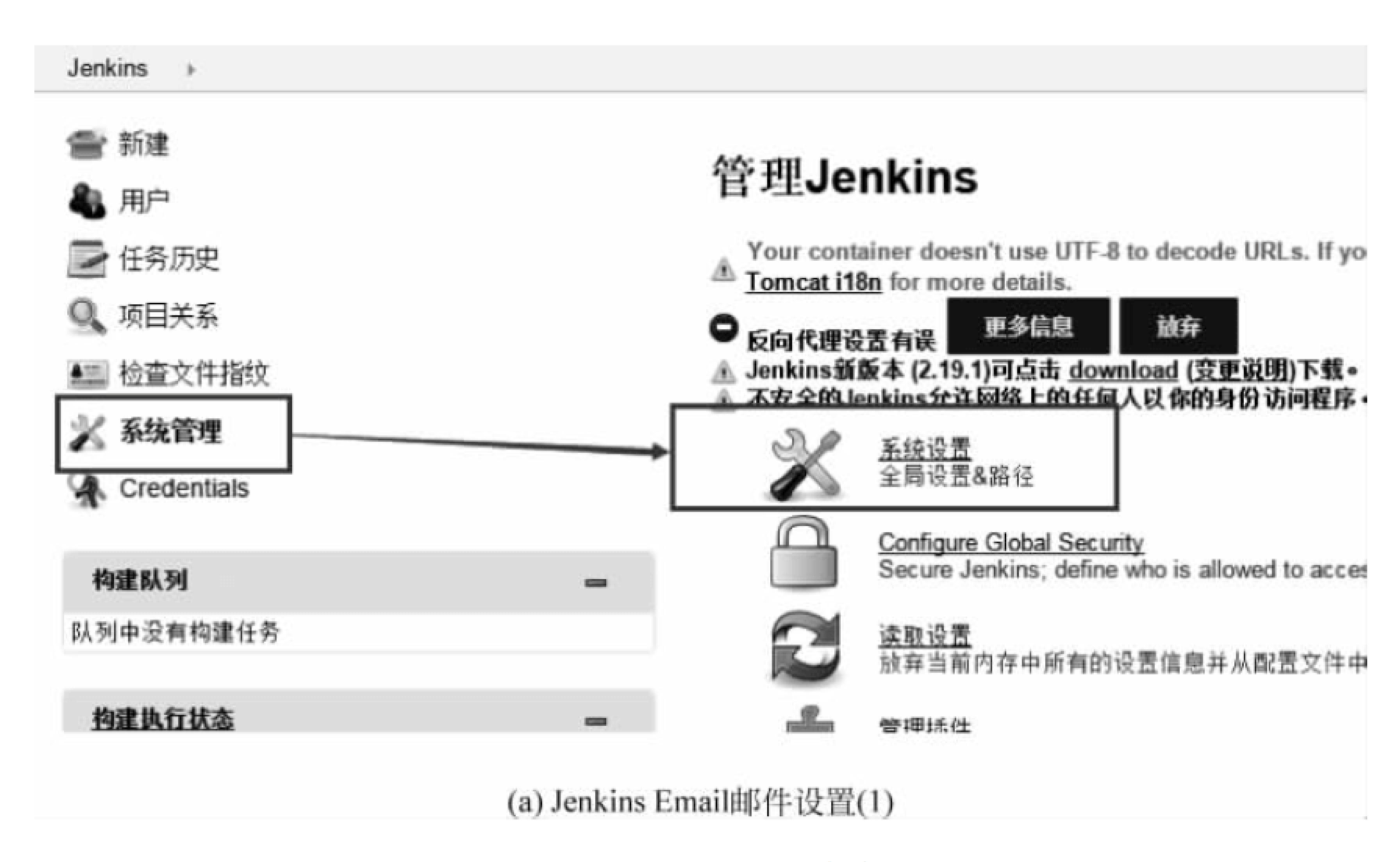
如果网站项目很多，人工去查看状态不可取，可以借助 Jenkins Email 插件实现网站构建完成，自动发送邮件给相应的开发人员、运维人员或者测试人员。Jenkins 发送邮件，需安装 Email 邮件插件：Email-ext、Token-macro 和 Email-template，Jenkins Email 邮件配置常见参数详解如下：

- SMTP server：邮件服务器地址。
- Default Content Type：内容展现的格式，一般选择 HTML。
- Default Recipients：默认收件人。
- Use SMTP Authentication：使用 SMTP 身份验证。
- User Name：邮件发送账户的用户名。
- Password：邮件发送账户的密码。
- SMTP port：SMTP 服务器端口。

Jenkins Email 邮件配置方法如下：

(1) 设置 Jenkins 邮件发送者，在 Jenkins 平台首页中选择"系统管理→系统设置→Jenkins Location"，填写 Jenkins URL 与系统管理员邮件地址，如图 22-23 所示。

(2) 设置发送邮件的 SMTP 服务器、邮箱后缀、发送类型 HTML、接收者或者抄送者，在 Jenkins 平台首页中选择"系统管理→系统设置→ Extended E-mail Notification"，填写如



(a) Jenkins Email邮件设置(1)

图 22-23　Jenkins Email 邮件设置

Maven项目配置
全局MAVEN_OPTS
Local Maven Repository Default (~/.m2/repository)
Help make Jenkins better by sending anonymous usage statistics and crash reports to the Jenkins project.
Jenkins Location
Jenkins URL http://121.42.183.93:7001/
系统管理员邮件地址 wuguangke@jfedu.net
SSH Server
SSHD Port ○指定端口： ◉随机选取 ○禁用

(b) Jenkins Email邮件设置(2)

图 22-23 （续）

图 22-24 所示的选项，包括 SMTP server、默认后缀、使用 SMTP 认证、Default Recipients 邮件接收人等信息。

Extended E-mail Notification
SMTP server mail.jfedu.net
Default user E-mail suffix @jfedu.net
Use SMTP Authentication
User Name wuguangke
Password ••••••••••
Use SSL
SMTP port
Charset UTF-8
Default Content Type HTML (text/html)
Use List-ID Email Header
Add 'Precedence: bulk' Email Header
Default Recipients wgkgood@163.com
Reply To List

图 22-24 Jenkins Email 邮件配置

（3）设置邮件的标题 Default Subject 内容如下：

构建通知：$ PROJECT_NAME -Build # $ BUILD_NUMBER - $ BUILD_STATUS

（4）设置发送邮件的内容，Default Content 内容如下：

```
<hr/>
<h3>(本邮件是程序自动下发的，请勿回复！)</h3><hr/>
项目名称： $PROJECT_NAME<br/><hr/>
构建编号： $BUILD_NUMBER<br/><hr/>
构建状态： $BUILD_STATUS<br/><hr/>
触发原因： ${CAUSE}<br/><hr/>
```

```
构建日志地址：<a href="${BUILD_URL}console">${BUILD_URL}console</a><br/><hr/>
构建地址：<a href="$BUILD_URL">$BUILD_URL</a><br/><hr/>
变更集：${JELLY_SCRIPT,template="html"}<br/>
<hr/>
```

（5）每个 JOB 工程邮件设置，单击 www.jfedu.net JOB 名称，选择"配置→构建后操作→Editable Email Notification"，如下信息保持默认，如图 22-25 所示。

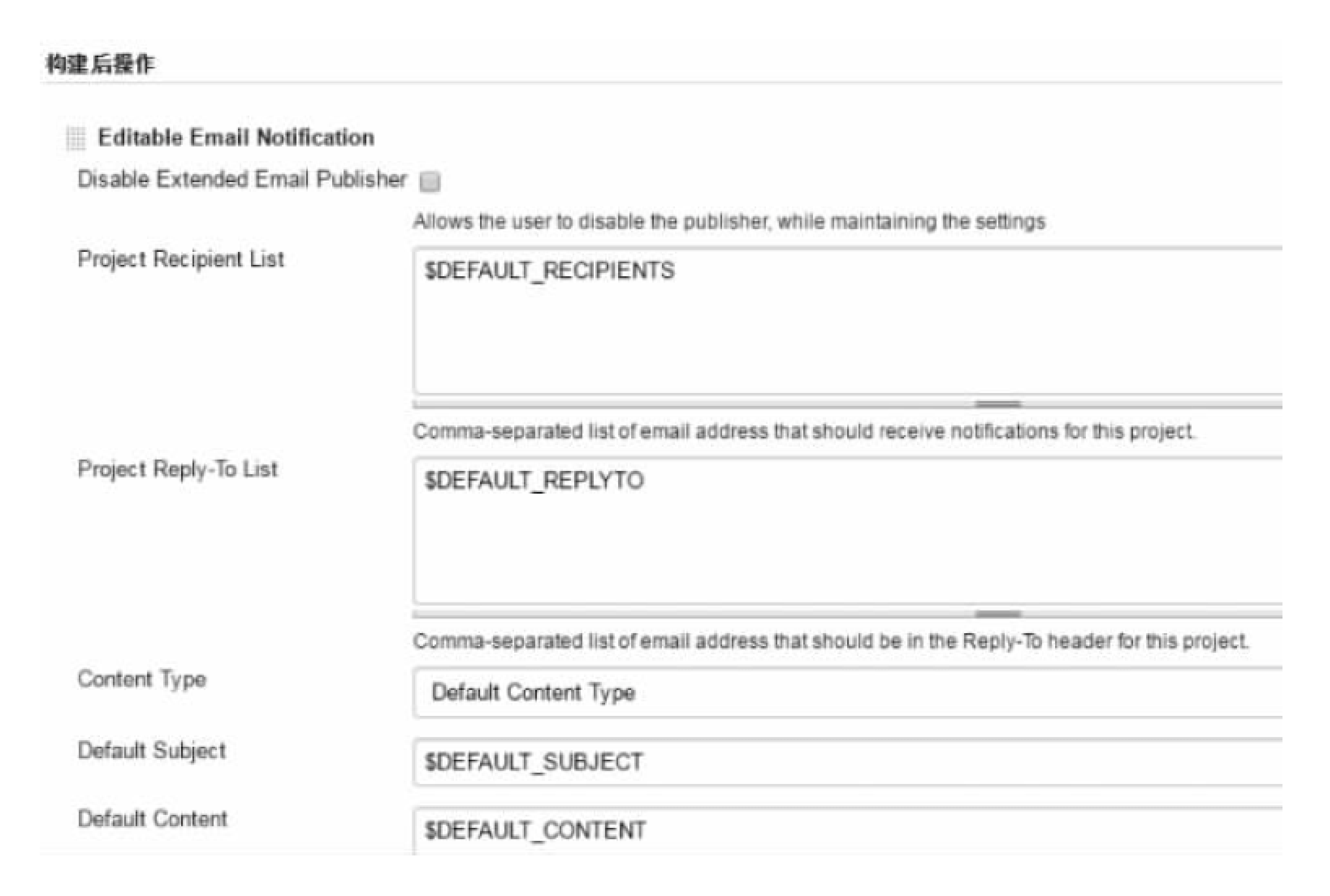


图 22-25 Jenkins Email JOB 邮件模板配置

（6）选择 Advanced Settings，设置 Trigger 阈值，选择发送邮件的触发器，默认触发器包括第一次构建、构建失败、总是发送邮件、构建成功等，一般选择 always 总是发送邮件，发送给 developers 组，如图 22-26 所示。



图 22-26 Jenkins Email 触发器设置

（7）Jenkins 构建邮件验证，如图 22-27 所示。

```
  已解压: doc/晒酷网-后台-1.2/resources/css/jquery-ui-themes.css
  已解压: doc/晒酷网-后台-1.2/start.html
  已解压: doc/TestResult.xlsx
  已解压: doc/悦分享_网络教育平台_1.0.2.docx
  已创建: META-INF/maven/
  已创建: META-INF/maven/com.shareku/
  已创建: META-INF/maven/com.shareku/edu/
  已解压: META-INF/maven/com.shareku/edu/pom.xml
  已解压: META-INF/maven/com.shareku/edu/pom.properties
+ cd /usr/local/tomcat2/
bash: 第 30 行:cd: /usr/local/tomcat2/: 没有那个文件或目录
+ rm -rf work
+ /bin/sh /usr/local/tomcat2//bin/startup.sh
/bin/sh: /usr/local/tomcat2//bin/startup.sh: 没有那个文件或目录
Build step 'Execute shell' marked build as failure
Archiving artifacts
Email was triggered for: Always
Sending email for trigger: Always
Sending email to: wgkgood@163.com
Finished: FAILURE
```

(a) Jenkins构建报错触发邮件

构建通知：www.jfedu.net - Build # 30 - Failure

发件人：hkgood<hkgood@jfedu.net>

收件人：[redacted]

时　间：2017年06月04日 21:42 (星期日)

(本邮件是程序自动下发的，请勿回复！)

项目名称：www.jfedu.net

构建编号：30

构建状态：Failure

触发原因：Started by user anonymous

(b) Jenkins Email邮件信息(1)

Build URL　http://139.224.227.121:7001/job/www.jfedu.net/30/

Project:　www.jfedu.net

Date of build:　Sun, 04 Jun 2017 21:41:07 +0800

Build duration:　1 分 50 秒

CHANGES

No Changes

BUILD ARTIFACTS

- target/edu.war

BUILD ARTIFACTS

edu Maven Webapp

- com.shareku/edu/0.0.1-SNAPSHOT/edu-0.0.1-SNAPSHOT.pom
- com.shareku/edu/0.0.1-SNAPSHOT/edu-0.0.1-SNAPSHOT.war

(c) Jenkins Email邮件信息(2)

图 22-27　Jenkins Email 邮件信息

22.12　Jenkins 多实例配置

单台 Jenkins 服务器可以满足企业测试及生产环境。如果每天更新发布多个 Web 网站，Jenkins 需要同时处理很多的任务。

基于 Jenkins 分布式，即多 slave 方式可以缓解 Jenkins 服务器的压力，Jenkins 多实例架构如图 22-28 所示，可以在 Windows、Linux、MAC 等操作系统上执行 slave。

Jenkins 多 slave 原理是将原本在 Jenkins master 端的构建项目分配给 slave 端去执行，Jenkins master 分配任务时，Jenkins master 端通过 SSH 远程 slave，在 slave 端启动 slave.jar 程序，通过 slave.jar 实现对网站工程的构建编译以及自动部署。所以在 slave 端服务器必须安装 Java JDK 环境来执行 master 端分配的构建任务。配置多 slave 服务器方法和步骤如下：

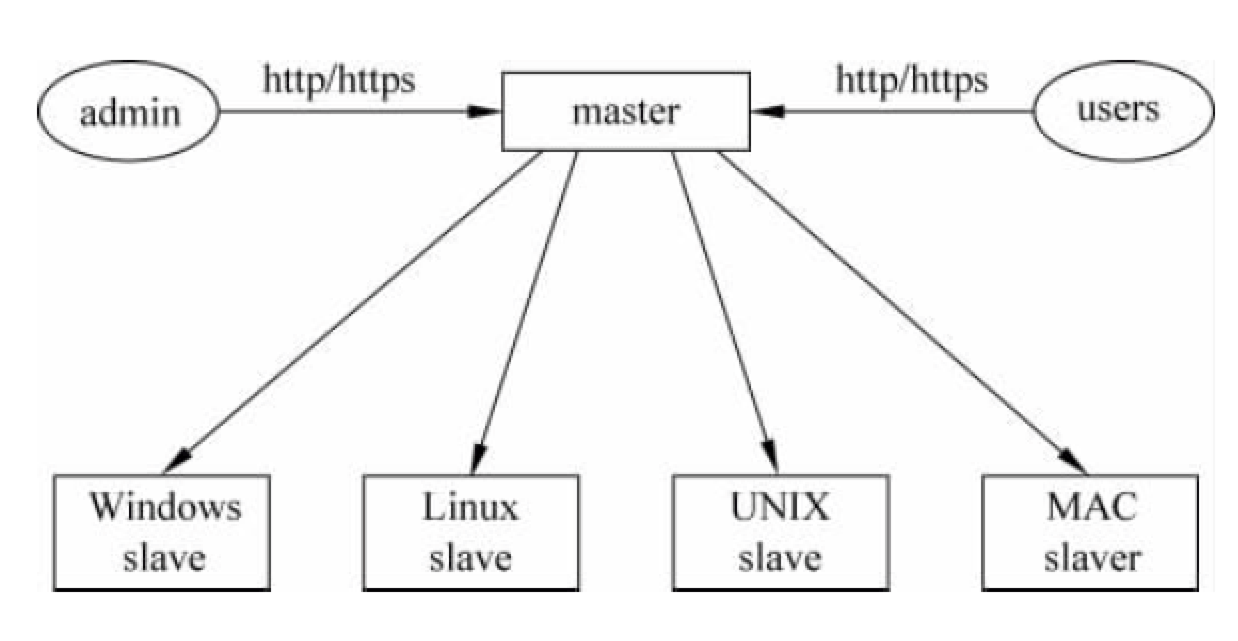


图 22-28　Jenkins slave 架构图

（1）在 slave 服务器，创建远程执行 Jenkins 任务的用户，名称为 jenkins，Jenkins 工作目录/home/Jenkins，Jenkins master 免秘钥登录 slave 服务器或者通过用户名和密码登录 slave。

（2）slave 服务器安装 Java JDK 版本，并将其软件路径加入系统环境变量。

（3）Jenkins master 端平台添加管理节点，依次选择"系统管理→管理节点→新建节点→输入节点名称"，如图 22-29 所示。



(a) 系统管理、管理节点

图 22-29　Jenkins slave 配置



(b) 新建节点

(c) 输入节点名称

图 22-29 （续）

（4）配置 www_slave 节点，指定其 Jenkins 编译工作目录，设置 IP 地址，在 Add Credentials 界面中添加登录 slave 用户名和密码，如图 22-30 所示。



(a) 配置www_slave节点

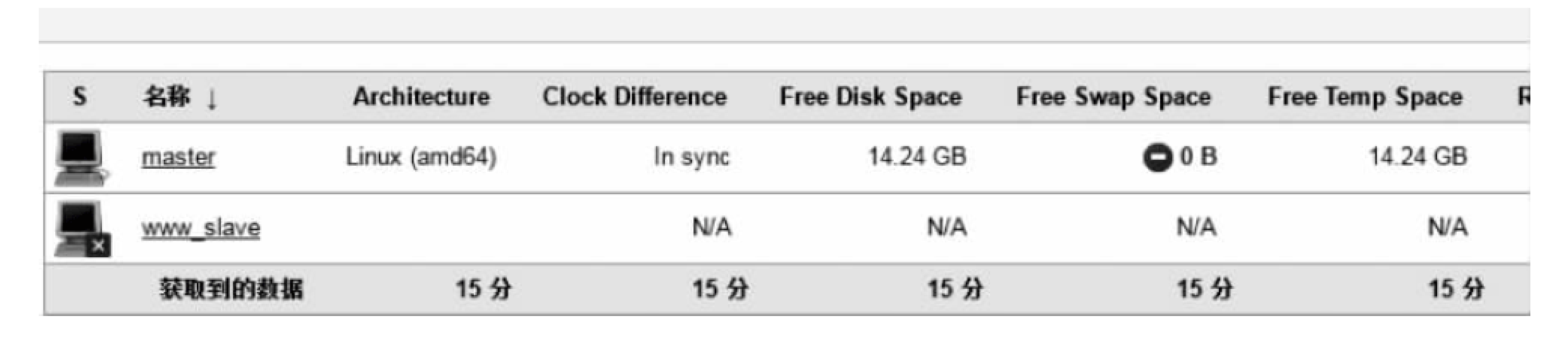
图 22-30 Jenkins slave 配置



(b) 添加登录slave用户名和密码

图 22-30 （续）

（5）Jenkins slave 配置完毕，查看 slave 状态如图 22-31 所示。



S	名称 ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	R
	master	Linux (amd64)	In sync	14.24 GB	0 B	14.24 GB	
	www_slave		N/A	N/A	N/A	N/A	
	获取到的数据	15 分	15 分	15 分	15 分	15 分	

图 22-31　Jenkins slave 状态信息

（6）单击 www_slave 节点，然后选择 Launch salve agent，单击测试 slave agent 是否正常工作，如图 22-32 所示。

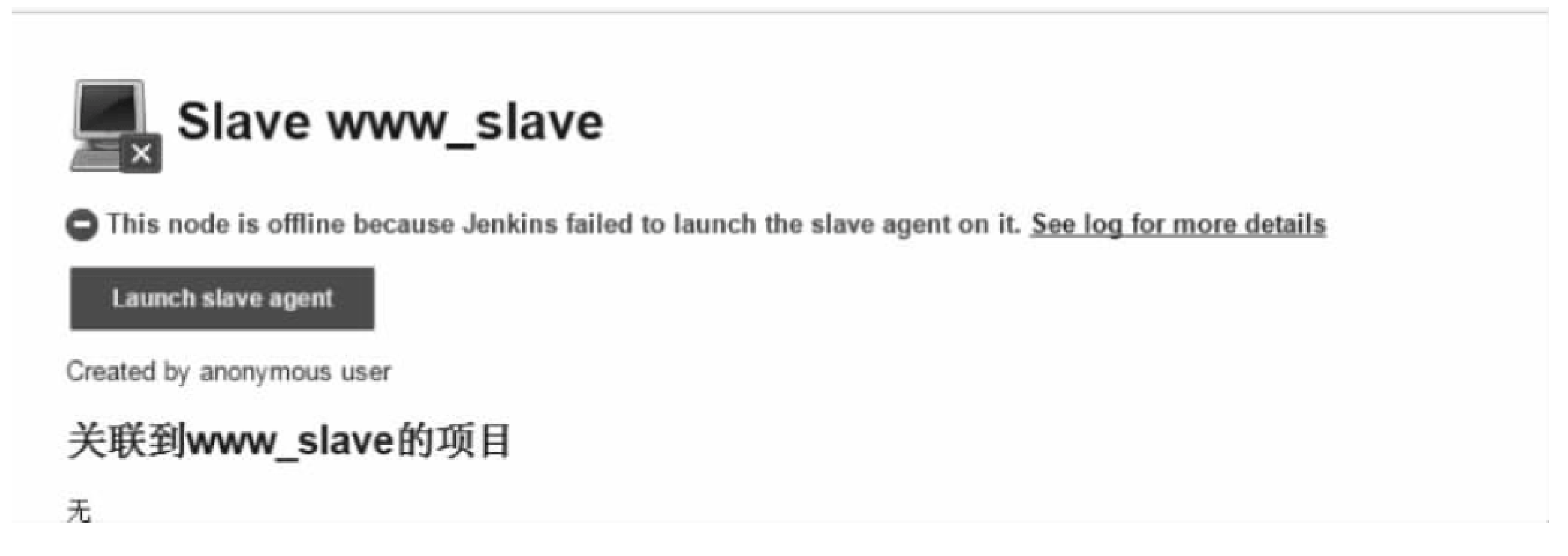


图 22-32　Jenkins slave agent 测试

（7）出现如图 22-33 所示的界面，即证明 slave 添加成功。

```
[01/08/17 16:34:48] [SSH] Opening SSH connection to 121.42.183.93:22.
[01/08/17 16:34:48] [SSH] Authentication successful.
[01/08/17 16:34:48] [SSH] The remote users environment is:
BASH=/bin/bash
BASHOPTS=cmdhist:extquote:force_fignore:hostcomplete:interactive_comments:progcomp:promptvars:sourcepath
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
BASH_CMDS=()
BASH_EXECUTION_STRING=set
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="4" [1]="1" [2]="2" [3]="1" [4]="release" [5]="x86_64-redhat-linux-gnu")
BASH_VERSION='4.1.2(1)-release'
CVS_RSH=ssh
DIRSTACK=()
EUID=0
GROUPS=()
G_BROKEN_FILENAMES=1
HOME=/root
HOSTNAME=BeiJing-JFEDU-NET-WEB-001.COM
HOSTTYPE=x86_64
ID=0
IFS=$' \t\n'
LANG=en_US.UTF-8
```

(a) Jenkins slave测试(1)

```
PIPESTATUS=([0]="0")
PPID=5325
PS4='+ '
PWD=/root
SHELL=/bin/bash
SHELLOPTS=braceexpand:hashall:interactive-comments
SHLVL=1
SSH_CLIENT='139.224.227.121 21604 22'
SSH_CONNECTION='139.224.227.121 21604 121.42.183.93 22'
TERM=dumb
UID=0
USER=root
_=/etc/bashrc
[01/08/17 16:34:48] [SSH] Starting sftp client.
[01/08/17 16:34:48] [SSH] Remote file system root /home/jenkins does not exist. Will try to create it...
[01/08/17 16:34:48] [SSH] Copying latest slave.jar...
[01/08/17 16:34:51] [SSH] Copied 522,364 bytes.
Expanded the channel window size to 4MB
[01/08/17 16:34:51] [SSH] Starting slave process: cd "/home/jenkins" && /usr/java/jdk1.7.0_25/bin/java  -jar slave.jar
<===[JENKINS REMOTING CAPACITY]===>channel started
Slave.jar version: 2.57
This is a Unix slave
```

(b) Jenkins slave测试(2)

图 22-33　Jenkins slave 测试

（8）如上配置完毕，Jenkins master 通过 SSH 方式来启动 slave 的 slave.jar 脚本，基于 Java 命令启动 slave.jar 包，命令为 java -jar slave.jar，slave 等待 master 端的任务分配，单击 www.jfedu.net，然后选择“立即构建”，如图 22-34 所示。

（9）Jenkins slave 配置完毕后，如果同时运行多个任务，会发现只会运行一个任务，另外的任务在等待，那需要怎么调整呢？此时需要配置 JOB 工程，选中“在必要的时候并发构建”即可，如图 22-35 所示。

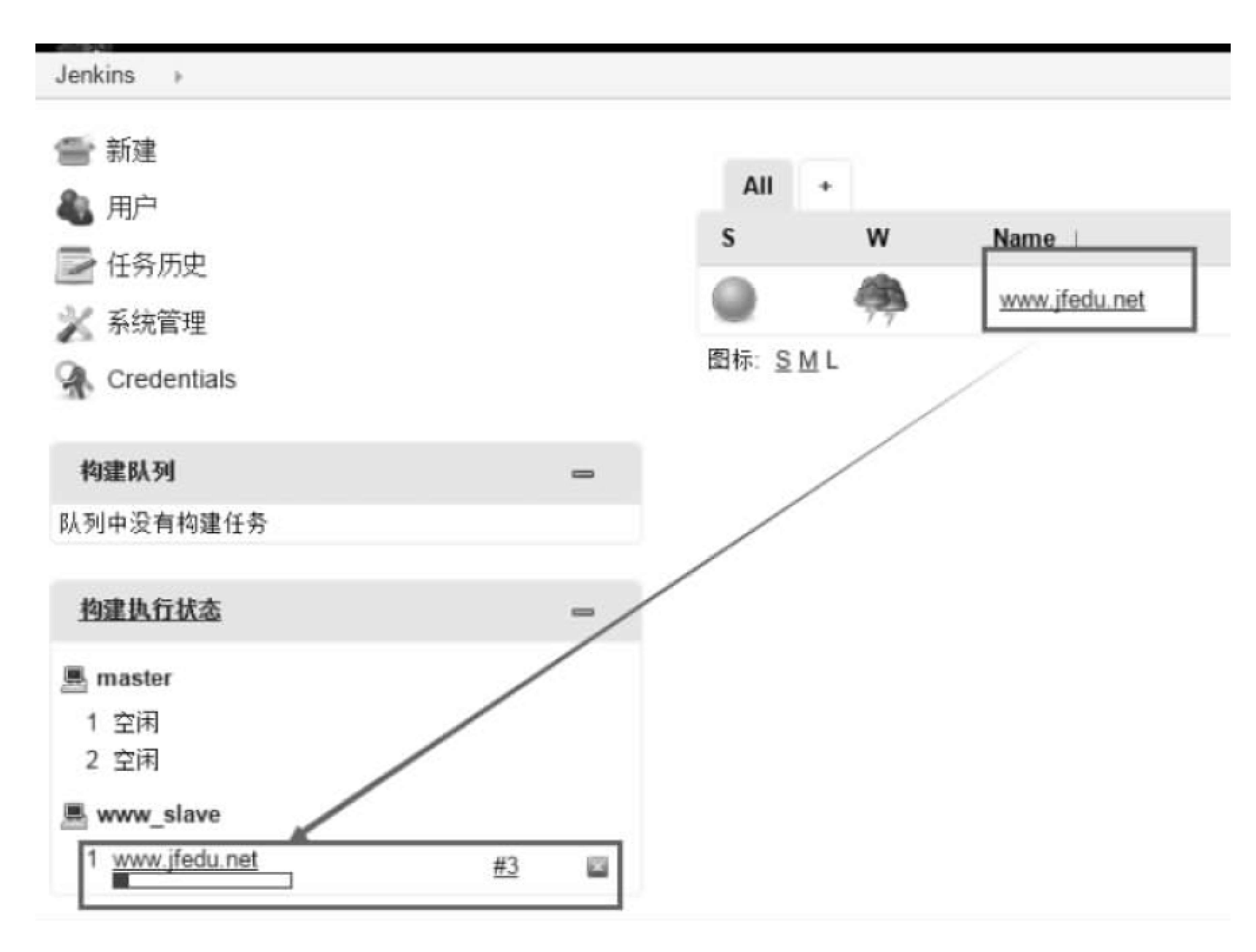


图 22-34　Jenkins slave 构建任务

项目名称　www.jfedu.net

描述　www.jfedu.net

[Plain text] 预览

- 丢弃旧的构建
- 参数化构建过程
- 关闭构建 (重新开启构建前不允许进行新的构建)
- 在必要的时候并发构建
- Restrict where this project can be run

高级项目选项

(a) 选中“在必要的时候并发构建”

图 22-35　Jenkins slave 构建多任务



(b) 多任务并发构建

图 22-35 （续）

22.13 Jenkins＋Ansible 高并发构建

Jenkins 自动部署基于 shell＋for 循环方式部署 10 台以下的 Java 服务器，效率是可以接受的，但是如果大规模服务器需要部署或者更新网站，通过 for 循环串行执行效率会大打折扣，所以需要考虑到并行机制。

Ansible 是一款极为灵活的开源工具套件，能够大大简化 UNIX 管理员的自动化配置管理与流程控制方式。它利用推送方式对客户系统加以配置，这样所有工作都可在主服务器端完成。使用 Ansible＋Jenkins 架构方式实现网站自动部署，满足上百台、千台服务器的网站部署和更新。

Ansible 服务需要部署在 Jenkins 服务器，客户端无须安装 Ansible。Ansible 基于 SSH 工作，所以需提前做好免秘钥或者通过 sudo 用户远程更新网站。此处省略 Ansible 安装，Ansible 相关知识请参考本书 Ansible 章节配置。

Ansible 自动部署网站有两种方法：一种是基于 Ansible 远程执行 shell 脚本；另外一种是 Ansible 编写 PlayBook 剧本，实现网站自动部署。以下为 Ansible＋shell 脚本方式自动部署网站方法。

（1）Jenkins 服务器安装 Ansible 软件，Red Hat、CentOS 操作系统可以直接基于 YUM 工具自动安装 Ansible，CentOS 6.X 或者 CentOS 7.X 安装 Ansible 前，需先安装 epel 扩展源，代码如下：

```
rpm -Uvh http://mirrors.ustc.edu.cn/fedora/epel/6/x86_64/epel-release-6-8.noarch.rpm
yum  install  epel-release  -y
yum  install  ansible       -y
```

（2）添加客户端服务器，在/etc/ansible/hosts 中添加需要部署的客户端 IP 列表，代码如下：

```
[www_jfedu]
139.199.228.59
139.199.228.60
139.199.228.61
139.199.228.62
```

（3）在 Jenkins 平台首页中单击 www.jfedu.net 项目，选择“配置→Post Steps→Execute shell→Command”，输入如下代码，www_jfedu 为 Ansible hosts 组模块名称。

```
cp /root/.jenkins/workspace/www.jfedu.net/target/edu.war /root/.jenkins/jobs/www.jfedu.
net/builds/lastSuccessfulBuild/archive/target/
ansible www_jfedu -m copy -a "src=/data/sh/auto_deploy.sh dest=/tmp/"
ansible www_jfedu -m shell -a "cd /tmp ; /bin/bash auto_deploy.sh"
```

（4）Jenkins 服务器端/data/sh/auto_deploy.sh shell 脚本内容如下：

```
#!/bin/bash
#Auto deploy Tomcat for jenkins
#By author jfedu.net 2017
export JAVA_HOME=/usr/java/jdk1.6.0_25
TOMCAT_PID='/usr/sbin/lsof -n -P -t -i :8081'
TOMCAT_DIR="/usr/local/tomcat/"
FILES="edu.war"
DES_DIR="/usr/local/tomcat/webapps/ROOT/"
DES_URL="http://139.224.227.121:7001/job/www.jfedu.net/lastSuccessfulBuild/artifact/
target/"
BAK_DIR="/export/backup/'date +%Y%m%d-%H%M'"
[ -n "$TOMCAT_PID" ] && kill -9 $TOMCAT_PID
cd $DES_DIR
rm -rf $FILES
mkdir -p $BAK_DIR; \cp -a $DES_DIR/* $BAK_DIR/
rm -rf $DES_DIR/*
wget $DES_URL/$FILES
/usr/java/jdk1.6.0_25/bin/jar -xvf $FILES
####################
cd $TOMCAT_DIR; rm -rf work
/bin/sh $TOMCAT_DIR/bin/start.sh
sleep 10
tail -n 50 $TOMCAT_DIR/logs/catalina.out
```

（5）单击 www.jfedu.net 构建任务，查看控制台信息，如图 22-36 所示。

```
[www.jfedu.net] $ /bin/sh -xe /usr/local/tomcat_jenkins/temp/hudson2957878034347171274.sh
+ cp /root/.jenkins/workspace/www.jfedu.net/target/edu.war /root/.jenkins/jobs/www.jfedu.net/builds/lastSuccessfu
+ ansible www_jfedu -m copy -a 'src=/data/sh/auto_deploy.sh dest=/tmp/'
/usr/lib/python2.6/site-packages/cryptography-1.7.1-py2.6-linux-x86_64.egg/cryptography/__init__.py:26: Deprecati
supported by the Python core team, please upgrade your Python. A future version of cryptography will drop support
  DeprecationWarning
139.199.228.59 | SUCCESS => {
    "changed": true,
    "checksum": "2ee998fb7ef482a498f1a312db913cce3e0d821b",
    "dest": "/tmp/auto_deploy.sh",
    "gid": 0,
    "group": "root",
    "md5sum": "5271b4d0917465453 53f79a96207a16b",
    "mode": "0644",
    "owner": "root",
    "size": 715,
    "src": "/root/.ansible/tmp/ansible-tmp-1496587744.11-232708497021134/source",
    "state": "file",
    "uid": 0
}
+ ansible www_jfedu -m shell -a 'cd /tmp ;/bin/bash auto_deploy.sh'
/usr/lib/python2.6/site-packages/cryptography-1.7.1-py2.6-linux-x86_64.egg/cryptography/__init__.py:26: Deprecati
supported by the Python core team, please upgrade your Python. A future version of cryptography will drop support
  DeprecationWarning
```

(a) Jenkins Ansible自动部署(1)

```
 24700K .......... .......... .......... .......... ..........  97%  246K 2s
 24750K .......... .......... .......... .......... ..........  98%  297K 2s
 24800K .......... .......... .......... .......... ..........  98%  291K 2s
 24850K .......... .......... .......... .......... ..........  98%  248K 1s
 24900K .......... .......... .......... .......... ..........  98%  297K 1s
 24950K .......... .......... .......... .......... ..........  98%  251K 1s
 25000K .......... .......... .......... .......... ..........  99%  362K 1s
 25050K .......... .......... .......... .......... ..........  99%  251K 1s
 25100K .......... .......... .......... .......... ..........  99%  247K 1s
 25150K .......... .......... .......... .......... ..........  99%  297K 0s
 25200K .......... .......... .......... .......... ..........  99%  296K 0s
 25250K .......... .......... .......... .......... ...        100%  259K=94s

2017-06-04 22:50:42 (270 KB/s) - 已保存 "edu.war" [25900686/25900686])
Archiving artifacts
Email was triggered for: Always
Sending email for trigger: Always
Sending email to: wgkgood@163.com
Finished: SUCCESS
```

(b) Jenkins Ansible自动部署(2)

139.199.228.59:7002

应用 Google 京峰教育-分布式监控 IBM developerWork Linux架构师【京峰教 京峰教育Linux运维

晒库网 shareku.com 晒你所长，与世界分享！ 购物车(0) 课程 请输入关键词

所有课程分类 首页 技能专区 考试考级 中小学 生活

IT/互联网、计算机

语言学习

中小学

你可以帮Ta的

不只有修电脑

(c) Jenkins Ansible自动部署(3)

图 22-36　Jenkins Ansible 自动部署

第 23 章 Linux 高可用集群实战



23.1 keepalived 高可用软件简介

目前互联网主流的实现 Web 网站及数据库服务高可用软件包括：keepalived、heartbeat 等。heartbeat 是比较早期的实现高可用软件，而 keepalived 是目前轻量级的管理方便、易用的高可用软件解决方案，得到互联网公司 IT 人的青睐。

keepalived 是一个类似于工作在 layer3、4 和 7 交换机制的软件，keepalived 软件有两种功能，分别是监控检查、VRRP 冗余协议。

keepalived 的作用是检测 Web 服务器的状态，如果有一台 Web 服务器、MySQL 服务器宕机或工作出现故障，keepalived 将检测到后，会将故障的 Web 服务器或者 MySQL 服务器从系统中剔除，当服务器工作正常后 keepalived 自动将 Web、MySQL 服务器加入到服务器群中，这些工作全部自动完成，不需要人工干涉，需要人工做的只是修复故障的 Web 和 MySQL 服务器。layer3、4 和 7 工作在 IP/TCP 协议栈的 IP 层、传输层及应用层，实现原理分别如下：

- layer3：keepalived 使用 layer3 的方式工作式时，keepalived 会定期向服务器群中的服务器发送一个 ICMP 的数据包，如果发现某台服务的 IP 地址无法 ping 通，keepalived 便报告这台服务器失效，并将它从服务器集群中剔除。layer3 的方式是以服务器的 IP 地址是否有效作为服务器工作正常与否的标准。
- layer4：layer4 主要以 TCP 端口的状态来决定服务器工作正常与否。如 Web server 的服务端口一般是 80，如果 keepalived 检测到 80 端口没有启动，则 keepalived 将把这台服务器从服务器群中剔除。
- layer7：layer7 工作在应用层，keepalived 将根据用户的设定检查服务器程序的运行是否正常，如果与用户的设定不相符，则 keepalived 将把服务器从服务器群中剔除。

23.2 keepalived VRRP 原理剖析

keepalived 是 VRRP 的完美实现，在学习 keepalived 之前，必须了解 VRRP 协议的原理。在现实的网络环境中，两台需要通信的主机大多数情况下并没有直接的物理连接。对

于这样的情况，它们之间的路由是怎样选择的呢？主机如何选定到达目的主机的下一跳路由，这个问题通常的解决方法有以下两种：

- 在主机上使用动态路由协议 RIP、OSPF；
- 在主机上配置静态路由。

在主机上配置动态路由是非常不切实际的，因为管理、维护成本以及是否支持等诸多问题。因此配置静态路由就变得十分流行，但路由器或者默认网关(default gateway)却经常成为单点，VRRP 的目的就是为了解决静态路由单点问题。VRRP 通过一竞选(election)协议来动态地将路由任务交给 LAN 中虚拟路由器中的某台 VRRP 路由器。

在 VRRP 虚拟路由器集群中，由多台物理的路由器组成，但是这多台的物理路由器并不能同时工作，而是由一台称为 master 路由器负责路由工作，其他的都是 backup，master 并非一成不变，VRRP 会让每个 VRRP 路由器参与竞选，最终获胜的就是 master。

Master 拥有一些特权，例如拥有虚拟路由器的 IP 地址，也即 VIP，拥有特权的 master 要负责转发给网关地址的包和响应 ARP 请求。

VRRP 通过竞选协议来实现虚拟路由器的功能，所有的协议报文都是通过 IP 多播(multicast)包(多播地址 224.0.0.18)形式发送的。虚拟路由器由 VRID(范围 0～255)和一组 IP 地址组成，对外表现为一个周知的 MAC 地址。所以在一组虚拟路由器集群中，不管谁是 master，对外都是相同的 MAC 和 VIP。客户端主机并不需要因为 master 的改变而修改自己的路由配置。

作为 master 的 VRRP 路由器会一直发送 VRRP 广播包(VRRP advertisement message)，backup 不会抢占 master，除非它的优先级(priority)更高。当 master 不可用时(backup 收不到广播包)，多台 backup 中优先级最高的这台会抢占为 master。这种抢占是非常快速的，以保证服务的连续性。从安全性考虑 VRRP 包使用了加密协议进行传输。

keepalived 基于 VRRP 技术，将两台物理主机当成路由器，两台物理机主机组成一个虚拟路由集群，master 高的主机产生 VIP，该 VIP 负责转发用户发起的 IP 包或者负责处理用户的请求，Nginx＋keepalived 组合，用户的请求直接访问 keepalived VIP 地址，然后访问 master 相应服务和端口。

23.3 企业级 Nginx＋keepalived 集群实战

随着 Nginx 在国内的发展潮流，越来越多的互联网公司都在使用 Nginx，Nginx 的高性能、稳定性成为 IT 人士青睐的 HTTP 和反向代理服务器。

Nginx 负载均衡一般位于整个网站架构的最前端或者中间层，如果为最前端时单台 Nginx 会存在单点故障，一台 Nginx 宕机，会影响用户对整个网站的访问。所以需要加入 Nginx 备份服务器，Nginx 主服务器与备份服务器之间形成高可用，一旦发现 Nginx 主宕机，能快速将网站切换至备份服务器。Nginx＋keepalived 网络架构如图 23-1 所示。

Nginx＋keepalived 高性能 Web 网络架构实战配置步骤如下。

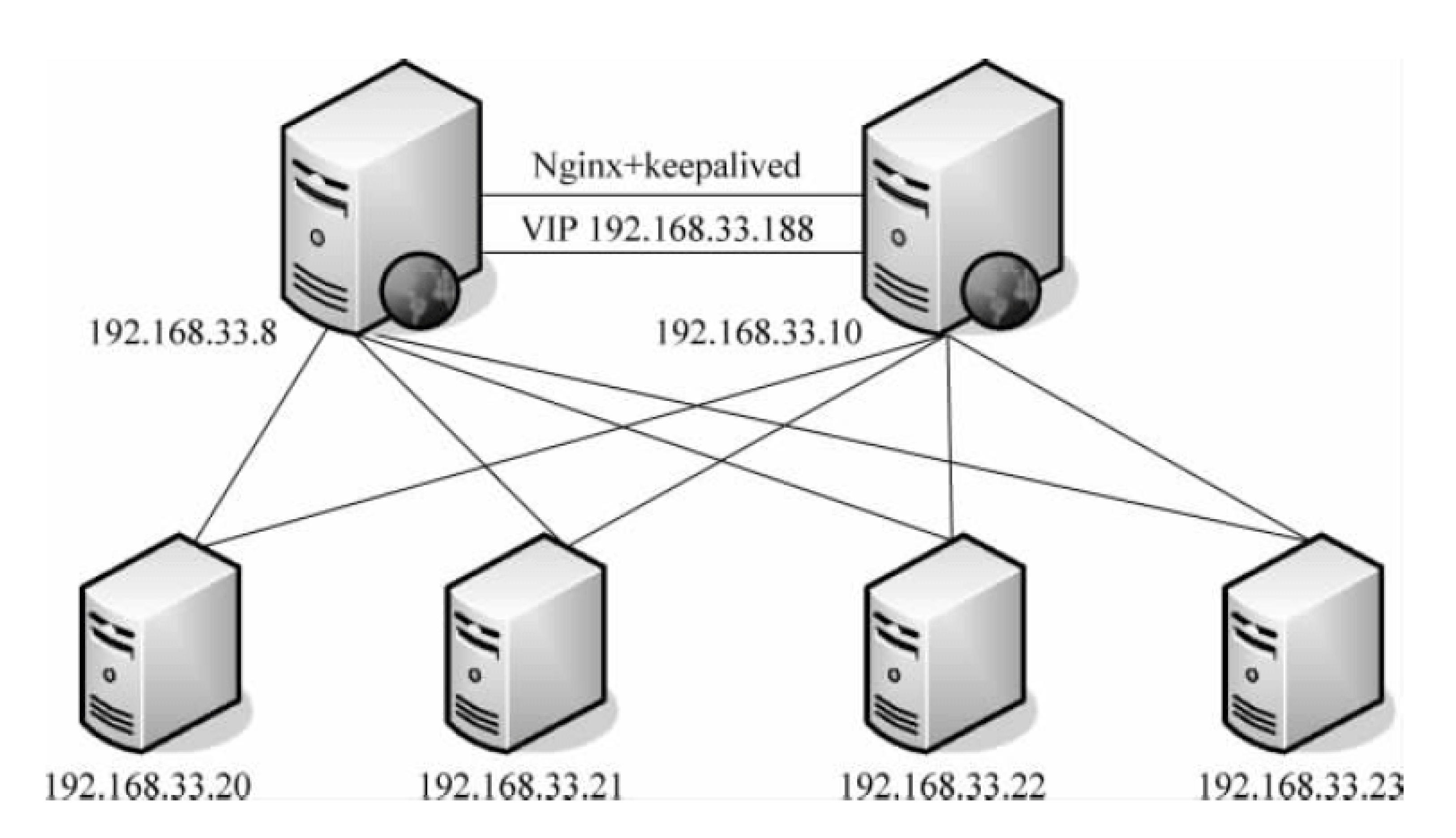


图 23-1 Nginx+keepalived 架构图

(1) 环境准备,具体内容如下:

- Nginx 版本:nginx v1.12.0。
- keepalived 版本:keepalived v1.2.1。
- Nginx-1:192.168.33.8(master)。
- Nginx-2:192.168.33.10(backup)。

(2) Nginx 安装配置,master 和 backup 服务器安装 Nginx、keepalived,执行命令 yum install -y pcre-devel 安装 Perl 兼容的正则表达式库,代码如下:

```
tar -xzf nginx-1.12.0.tar.gz
cd nginx-1.12.0
sed -i -e 's/1.12.0//g' -e 's/nginx\//TDTWS/g' -e 's/"NGINX"/"TDTWS"/g' src/core/nginx.h
./configure --prefix=/usr/local/nginx --user=www --group=www --with-http_stub_
status_module --with-http_ssl_module
make
make install
```

(3) keepalived 安装配置,代码如下:

```
tar -xzvf keepalived-1.2.1.tar.gz
cd keepalived-1.2.1
./configure
make
make install
DIR=/usr/local/
cp $DIR/etc/rc.d/init.d/keepalived /etc/rc.d/init.d/
cp $DIR/etc/sysconfig/keepalived   /etc/sysconfig/
mkdir -p /etc/keepalived
cp    $DIR/sbin/keepalived         /usr/sbin/
```

（4）配置 keepalived，两台服务器 keepalived.conf 内容都配置如下，state 均设置为 BACKUP，backup 服务器需要修改优先级为 90，代码如下：

```
! Configuration File for keepalived
 global_defs {
  notification_email {
      support@jfedu.net
 }
    notification_email_from wgkgood@163.com
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
 }

 vrrp_script chk_nginx {
    script "/data/sh/check_nginx.sh"
    interval 2
    weight 2
 }
 # VIP1
 vrrp_instance VI_1 {
     state BACKUP
     interface eth0
     lvs_sync_daemon_inteface eth0
     virtual_router_id 151
     priority 100
     advert_int 5
     nopreempt
     authentication {
         auth_type PASS
         auth_pass 1111

     }
     virtual_ipaddress {
         192.168.33.188
     }
     track_script {
     chk_nginx
    }
 }
```

如上配置还需要建立 check_nginx 脚本，用于检查本地 Nginx 是否存活，如果不存活，则 stop keepalived 服务实现切换。其中 check_nginx.sh 脚本内容如下：

```
#!/bin/bash
#auto check nginx process
#2017-5-26 17:47:12
```

```
#by author jfedu.net
killall -0   nginx
if [[ $? -ne 0 ]]; then
    /etc/init.d/keepalived stop
fi
```

在两台 Nginx 服务器发布目录分别新建 index.html 测试页面，然后启动 Nginx 服务测试，访问 VIP 地址，即访问 http://192.168.33.188 即可。

23.4　企业级 Nginx＋keepalived 双主架构实战

Nginx＋keepalived 主备模式，始终存在一台服务器处于空闲状态，如何更好地把两台服务器利用起来呢，可以借助 Nginx＋keepalived 双主架构来实现，如图 23-2 所示，将架构改成 Nginx 双主架构，同时两台对外提供服务，拥有两个 VIP 地址，同时接收用户的请求。

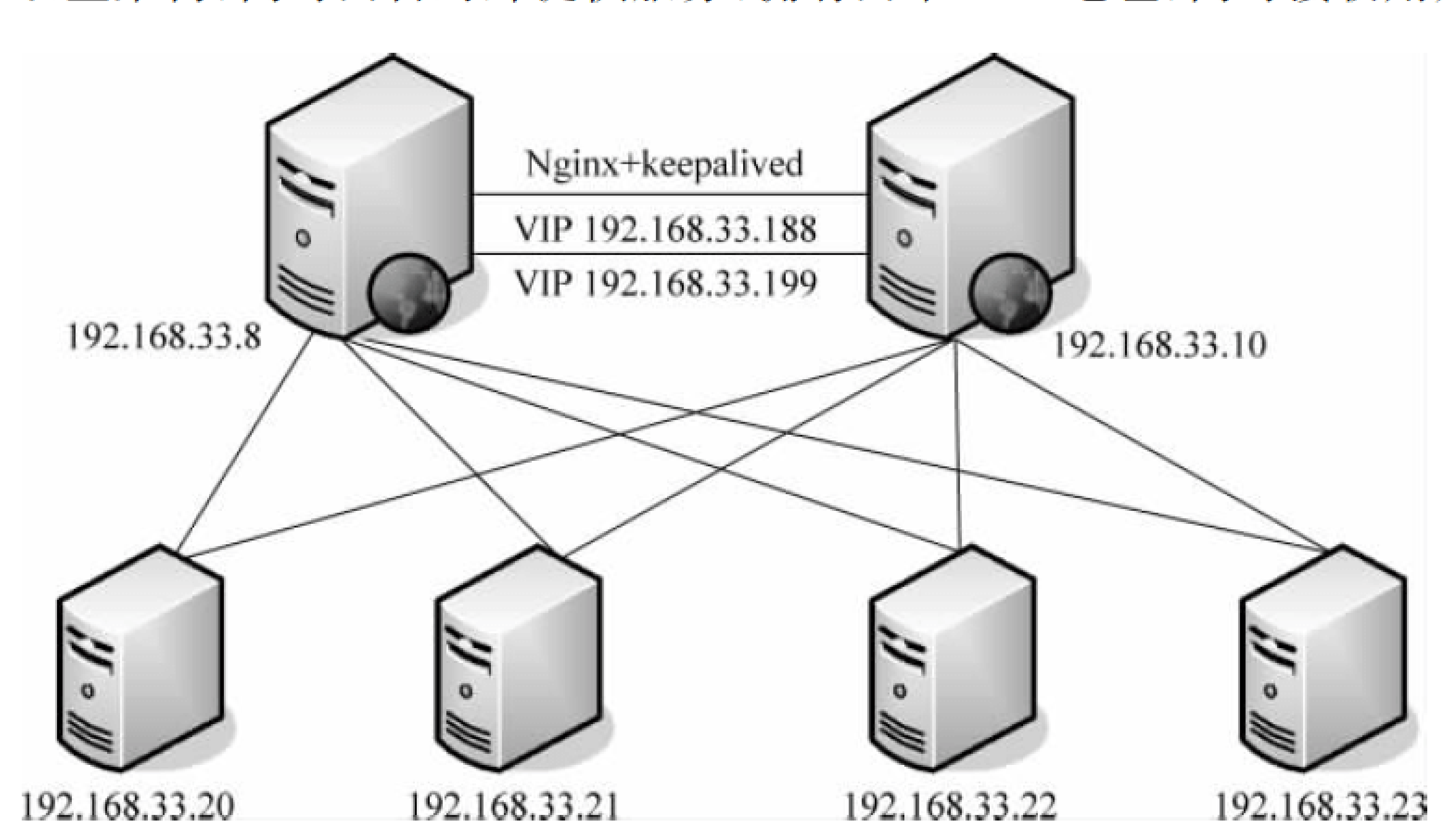


图 23-2　Nginx＋keepalived 双主架构

Nginx＋keepalived 双主架构实现方法步骤如下：

（1）master1 上 keepalived.conf 配置文件内容如下：

```
! Configuration File for keepalived
 global_defs {
  notification_email {
      wgkgood@163.com
 }
    notification_email_from wgkgood@163.com
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
 }
 vrrp_script chk_nginx {
```

```
        script "/data/sh/check_nginx.sh"
        interval 2
        weight 2
    }
    # VIP1
    vrrp_instance VI_1 {
        state MASTER
        interface eth0
        lvs_sync_daemon_inteface eth0
        virtual_router_id 151
        priority 100
        advert_int 5
        nopreempt
        authentication {
            auth_type PASS
            auth_pass 1111
        }
        virtual_ipaddress {
            192.168.33.188
        }
        track_script {
        chk_nginx
       }
    }
   # VIP2
    vrrp_instance VI_2 {
        state BACKUP
        interface eth0
        lvs_sync_daemon_inteface eth0
        virtual_router_id 152
        priority 90
        advert_int 5
        nopreempt
        authentication {
            auth_type PASS
            auth_pass 2222
        }
        virtual_ipaddress {
            192.168.33.199
        }
        track_script {
        chk_nginx
       }
    }
```

（2）master2 上 keepalived.conf 配置文件内容如下：

```
! Configuration File for keepalived
 global_defs {
  notification_email {
      wgkgood@163.com
 }
    notification_email_from wgkgood@163.com
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
 }
 vrrp_script chk_nginx {
    script "/data/sh/check_nginx.sh"
    interval 2
    weight 2
 }
 # VIP1
 vrrp_instance VI_1 {
     state BACKUP
     interface eth0
     lvs_sync_daemon_inteface eth0
     virtual_router_id 151
     priority 90
     advert_int 5
     nopreempt
     authentication {
         auth_type PASS
         auth_pass 1111
     }
     virtual_ipaddress {
         192.168.33.188
     }
     track_script {
     chk_nginx
    }
 }
# VIP2
 vrrp_instance VI_2 {
     state MASTER
     interface eth0
     lvs_sync_daemon_inteface eth0
     virtual_router_id 152
     priority 100
     advert_int 5
```

```
        nopreempt
        authentication {
            auth_type PASS
            auth_pass 2222
        }
        virtual_ipaddress {
            192.168.33.199
        }
        track_script {
        chk_nginx
       }
   }
```

(3) 两台 Nginx 服务器上配置/data/sh/check_nginx.sh 脚本，内容如下：

```
#!/bin/bash
#auto check nginx process
killall - 0   nginx
if
[[  $? - ne 0 ]]; then
/etc/init.d/keepalived stop
fi
```

(4) 如图 23-3 所示，两个 VIP 在一台服务器，由于另外一台服务器 DOWN 机，VIP 都漂移到本机网卡下。

```
[root@localhost ~]# ip addr list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:16:3e:00:16:58 brd ff:ff:ff:ff:ff:ff
    inet 10.171.60.52/21 brd 10.171.63.255 scope global eth0
    inet 192.168.33.188/32 scope global eth0
    inet 192.168.33.199/32 scope global eth0
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:16:3e:00:13:4c brd ff:ff:ff:ff:ff:ff
    inet 182.92.188.163/22 brd 182.92.191.255 scope global eth1
[root@localhost ~]#
```

图 23-3 VIP 漂移在单台服务器

(5) Nginx+keepalived 双主企业架构，在日常维护及管理过程中需要以下几个方面：

- keepalived 主配置文件必须设置不同的 VRRP 名称，同时优先级和 VIP 设置也各不相同。
- Nginx 网站总访问量为两台 Nginx 服务器访问之和，可以写脚本自动统计访问量。
- 两台 Nginx 为 master，存在两个 VIP 地址，用户从外网访问 VIP，需配置域名映射到两个 VIP 上方即可。
- 通过外网 DNS 映射不同 VIP 的方法也称为 DNS 负载均衡模式。
- 可以通过 Zabbix 实时监控 VIP 访问状态是否正常。

23.5 Redis+keepalived 高可用集群实战

Redis 主从复制优点及应用场景，Web 应用程序可以基于主从同步实现读写分离以提高服务器的负载能力。

在常见的场景中，读的频率一般比较大，当单机 Redis 无法应对大量的读请求时，可以通过复制功能建立多个从数据库，主数据库只进行写操作，而从数据库负责读操作，基于 Redis+keepalived 对 Redis 实现高可用，保证网站正常访问。以下为 Redis+keepalived 高可用架构实现步骤。

(1) Redis 主库、从库分别安装 keepalived 服务，代码如下：

```
tar -xzvf keepalived-1.2.1.tar.gz
cd keepalived-1.2.1
./configure
make
make install
DIR=/usr/local/
cp $DIR/etc/rc.d/init.d/keepalived /etc/rc.d/init.d/
cp $DIR/etc/sysconfig/keepalived   /etc/sysconfig/
mkdir -p /etc/keepalived
cp    $DIR/sbin/keepalived          /usr/sbin/
```

(2) Redis+keepalived master 配置文件代码如下：

```
! Configuration File for keepalived
 global_defs {
  notification_email {
      wgkgood@163.com
 }
    notification_email_from wgkgood@163.com
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
 }
 vrrp_script chk_redis {
    script "/data/sh/check_redis.sh"
    interval 2
    weight 2
 }
 # VIP1
 vrrp_instance VI_1 {
     state MASTER
     interface eth0
     lvs_sync_daemon_inteface eth0
     virtual_router_id 151
```

```
        priority 100
        advert_int 5
        nopreempt
        authentication {
            auth_type PASS
            auth_pass 1111
        }
        virtual_ipaddress {
            192.168.33.188
        }
        track_script {
        chk_redis
       }
}
```

（3）Redis+keepalived backup 配置文件代码如下：

```
! Configuration File for keepalived
 global_defs {
  notification_email {
      wgkgood@163.com
 }
    notification_email_from wgkgood@163.com
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
 }
 vrrp_script chk_redis {
    script "/data/sh/check_redis.sh"
    interval 2
    weight 2
 }
 # VIP1
 vrrp_instance VI_1 {
        state BACKUP
        interface eth0
        lvs_sync_daemon_inteface eth0
        virtual_router_id 151
        priority 90
        advert_int 5
        nopreempt
        authentication {
            auth_type PASS
            auth_pass 1111
        }
        virtual_ipaddress {
            192.168.33.188
```

```
        }
        track_script {
        chk_redis
       }
 }
```

（4）两台 Redis 服务器上配置/data/sh/check_redis.sh 脚本，内容如下：

```
#!/bin/bash
#auto check redis process
NUM='ps -ef |grep redis|grep -v grep|grep -v check|wc -l'
if
[[ $NUM -eq 0 ]];then
        /etc/init.d/keepalived stop
fi
```

23.6 NFS+keepalived 高可用集群实战

NFS(network file system)是一个网络文件系统，是 Linux 系统直接支持文件共享的一个文件系统，它允许网络中的计算机之间通过 TCP/IP 网络共享资源。在 NFS 的应用中，本地 NFS 的客户端应用可以透明地读写位于远端 NFS 服务器上的文件，就像访问本地文件一样。一般 NFS 为单机部署，而 NFS 服务器主要用于存放企业重要数据，此时为了保证数据的安全可靠，需要对 NFS 服务器之间实现同步。NFS+keepalived 高可用满足企业业务需求，以下为 NFS+keepalived 高可用架构实现步骤。

（1）NFS 主和备分别安装 keepalived 服务，代码如下：

```
tar -xzvf keepalived-1.2.1.tar.gz
cd keepalived-1.2.1
./configure
make
make install
DIR=/usr/local/
cp $DIR/etc/rc.d/init.d/keepalived /etc/rc.d/init.d/
cp $DIR/etc/sysconfig/keepalived   /etc/sysconfig/
mkdir -p /etc/keepalived
cp   $DIR/sbin/keepalived         /usr/sbin/
```

（2）NFS+keepalived master 配置文件代码如下：

```
! Configuration File for keepalived
 global_defs {
  notification_email {
      wgkgood@163.com
 }
    notification_email_from wgkgood@163.com
```

```
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
}
vrrp_script chk_nfs {
    script "/data/sh/check_nfs.sh"
    interval 2
    weight 2
}
# VIP1
vrrp_instance VI_1 {
     state MASTER
     interface eth0
     lvs_sync_daemon_inteface eth0
     virtual_router_id 151
     priority 100
     advert_int 5
     nopreempt
     authentication {
          auth_type PASS
          auth_pass 1111
     }
     virtual_ipaddress {
          192.168.33.188
     }
     track_script {
     chk_nfs
    }
}
```

（3）NFS+keepalived backup 配置文件代码如下：

```
! Configuration File for keepalived
global_defs {
  notification_email {
       wgkgood@163.com
}
    notification_email_from wgkgood@163.com
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
}
vrrp_script chk_nfs {
    script "/data/sh/check_nfs.sh"
    interval 2
    weight 2
}
```

```
# VIP1
vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    lvs_sync_daemon_inteface eth0
    virtual_router_id 151
    priority 90
    advert_int 5
    nopreempt
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.33.188
    }
    track_script {
    chk_nfs
   }
}
```

(4) 两台 NFS 服务器上配置/data/sh/check_nfs.sh 脚本,内容如下:

```
#!/bin/bash
#auto check nfs process
NUM = 'ps  - ef |grep nfs|grep  - v grep|grep  - v check|wc  - l'
if
[[  $NUM  - eq 0 ]];then
        /etc/init.d/keepalived stop
```

23.7 MySQL+keepalived 高可用集群实战

MySQL 主从同步复制可以实现取数据库进行备份,保证网站数据的快速恢复,一般应用程序读写均在 master 上,如果一旦 master 服务器宕机,需要手工切换 Web 网站连接数据库的 IP 至从库,可以基于 keepalived 软件实现自动 IP 切换,保证网站高可用率,MySQL+keepalived 集群架构配置方法如下。

(1) MySQL 主库、从库分别安装 keepalived 服务,代码如下:

```
tar  - xzvf keepalived - 1.2.1.tar.gz
cd keepalived - 1.2.1
./configure
make
make install
DIR = /usr/local/
cp  $DIR/etc/rc.d/init.d/keepalived    /etc/rc.d/init.d/
```

```
cp $DIR/etc/sysconfig/keepalived   /etc/sysconfig/
mkdir -p /etc/keepalived
cp    $DIR/sbin/keepalived          /usr/sbin/
```

（2）MySQL+keepalived master 配置文件代码如下：

```
! Configuration File for keepalived
 global_defs {
  notification_email {
      wgkgood@163.com
 }
   notification_email_from wgkgood@163.com
   smtp_server 127.0.0.1
   smtp_connect_timeout 30
   router_id LVS_DEVEL
 }
 vrrp_script chk_mysql {
   script "/data/sh/check_mysql.sh"
   interval 2
   weight 2
 }
 # VIP1
 vrrp_instance VI_1 {
     state MASTER
     interface eth0
     lvs_sync_daemon_inteface eth0
     virtual_router_id 151
     priority 100
     advert_int 5
     nopreempt
     authentication {
         auth_type PASS
         auth_pass 1111
     }
     virtual_ipaddress {
         192.168.33.188
     }
     track_script {
     chk_mysql
    }
 }
```

（3）MySQL+keepalived backup 配置文件代码如下：

```
! Configuration File for keepalived
 global_defs {
  notification_email {
      wgkgood@163.com
 }
   notification_email_from wgkgood@163.com
```

```
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
}
vrrp_script chk_mysql {
    script "/data/sh/check_mysql.sh"
    interval 2
    weight 2
}
# VIP1
vrrp_instance VI_1 {
     state BACKUP
     interface eth0
     lvs_sync_daemon_inteface eth0
     virtual_router_id 151
     priority 90
     advert_int 5
     nopreempt
     authentication {
          auth_type PASS
          auth_pass 1111
     }
     virtual_ipaddress {
          192.168.33.188
     }
     track_script {
     chk_mysql
    }
}
```

（4）两台 MySQL 服务器上配置/data/sh/check_mysql.sh 脚本，内容如下：

```
#!/bin/bash
#auto check mysql process
NUM='ps -ef |grep mysql|grep -v grep|grep -v check|wc -l'
if
[[ $NUM -eq 0 ]];then
        /etc/init.d/keepalived stop
fi
```

（5）Web 网站直接连接 keepalived VIP 即可实现网站自动切换，发现 MySQL 宕机，会自动切换至从库上。

23.8　Haproxy+keepalived 高可用集群实战

随着互联网火热的发展，开源负载均衡器的大量应用，企业主流软件负载均衡如 LVS、Haproxy、Nginx 等，各方面性能不亚于硬件负载均衡 F5，Haproxy 提供高可用性、负载均

衡以及基于 TCP 和 HTTP 应用的代理，支持虚拟主机，它是免费、快速并且可靠的一种解决方案。

23.8.1 Haproxy 入门简介

Haproxy 特别适用于那些负载特大的 Web 站点，这些站点通常又需要会话保持或七层处理。负载均衡 LVS 是基于四层，新型的大型互联网公司也在采用 Haproxy，了解了 Haproxy 大并发、七层应用等，Haproxy 高性能负载均衡优点如下：

- Haproxy 是支持虚拟主机的，可以工作在 4、7 层；
- 能够补充 Nginx 的一些缺点，比如 Session 的保持、Cookie 的引导等工作；
- 支持 url 检测后端的服务器；
- 它跟 LVS 一样，只是一款负载均衡软件，单纯从效率上来讲，Haproxy 更会比 Nginx 有更出色的负载均衡速度，在并发处理上也是优于 Nginx 的；
- Haproxy 可以对 MySQL 读进行负载均衡，对后端的 MySQL 节点进行检测和负载均衡，Haproxy 支持多种算法。

Haproxy+keepalived 企业高性能 Web 能够支持千万级并发网站，实现 Haproxy 高性能 Web 网站架构配置步骤如下。

23.8.2 Haproxy 安装配置

Haproxy 安装配置步骤相对比较简单，跟其他源码软件安装方法大致相同，以下为 Haproxy 配置方法及步骤。

(1) Haproxy 编译及安装，代码如下：

```
cd /usr/src
wget http://haproxy.1wt.eu/download/1.4/src/haproxy-1.4.21.tar.gz
tar xzf haproxy-1.4.21.tar.gz
cd haproxy-1.4.21
make TARGET=linux26 PREFIX=/usr/local/haproxy/
make install PREFIX=/usr/local/haproxy
```

(2) 配置 Haproxy 服务，代码如下：

```
cd /usr/local/haproxy ;mkdir -p etc/
touch /usr/local/haproxy/etc/haproxy.cfg
```

(3) haproxy.cfg 配置文件内容如下：

```
global
    log 127.0.0.1 local0
    log 127.0.0.1 local1 notice
    maxconn 4096
    uid 99
    gid 99
```

```
    daemon
defaults
    log global
    mode http
    option httplog
    option dontlognull
    retries 3
    option redispatch
    maxconn 2000
    contimeout 5000
    clitimeout 50000
    srvtimeout 50000
frontend http-in
    bind *:80
    acl is_www.jf1.com hdr_end(host) -i jf1.com
    acl is_www.jf2.com hdr_end(host) -i jf2.com
    use_backend www.jf1.com if is_www.jf1.com
    use_backend www.jf2.com if is_www.jf2.com
    default_backend www.jf1.com
backend www.jf1.com
    balance roundrobin
    cookie SERVERID insert nocache indirect
    option httpchk HEAD /index.html HTTP/1.0
    option httpclose
    option forwardfor
    server jf1 192.168.33.11:80 cookie jf1 check inter 1500 rise 3 fall 3 weight 1
backend www.jf2.com
    balance roundrobin
    cookie SERVERID insert nocache indirect
    option httpchk HEAD /index.html HTTP/1.0
    option httpclose
    option forwardfor
    server jf2 192.168.33.11:81 cookie jf2 check inter 1500 rise 3 fall 3 weight 1
```

（4）启动 Haproxy 服务，代码如下：

```
/usr/local/haproxy/sbin/haproxy   -f   /usr/local/haproxy/etc/haproxy.cfg
```

启动 Haproxy 报错如下：

```
[WARNING] 217/202150 (2857) : Proxy 'chinaapp.sinaapp.com': in multi-process mode, stats will be limited to process assigned to the current request.
```

修改源码配置 src/cfgparse.c 找到如下行，调整 nbproc > 1 数值即可：

```
if (nbproc > 1) {
                if (curproxy->uri_auth) {
-                       Warning("Proxy '%s': in multi-process mode, stats will be limited to
```

```
process assigned to the current request.\n",
+                    Warning("Proxy '%s': in multi-process mode, stats will be limited to
the process assigned to the current request.\n",
```

23.8.3 Haproxy 配置文件详解

Haproxy 配置文件内容详解如下：

```
######全局配置信息#######
global
        maxconn 20480                   #默认最大连接数
        log 127.0.0.1 local3            #[err warning info debug]
        chroot /usr/local/haproxy       #chroot 运行的路径
        uid 99                          #所属运行的用户 uid
        gid 99                          #所属运行的用户组
        daemon                          #以后台形式运行 haproxy
        nbproc 8                        #进程数量(可以设置多个进程提高性能)
        pidfile /usr/local/haproxy/haproxy.pid  #haproxy 的 pid 存放路径,启动进程的用户必
                                                #须有权限访问此文件
        ulimit-n 65535                  #ulimit 的数量限制
###########默认的全局设置##########
##这些参数可以被利用配置到 frontend,backend,listen 组件##
defaults
        log global
        mode http                       #所处理的类别(#7 层 http;4 层 tcp )
        maxconn 20480                   #最大连接数
        option httplog                  #日志类别 http 日志格式
        option httpclose                #每次请求完毕后主动关闭 http 通道
        option dontlognull              #不记录健康检查的日志信息
        option forwardfor               #如果后端服务器需要获得客户端真实 ip 需要配置的参
                                        #数,可以从 Http Header 中获得客户端 ip
        option redispatch               #serverId 对应的服务器挂掉后,强制定向到其他健康
                                        #的服务器
        option abortonclose             #当服务器负载很高的时候,自动结束掉当前队列处理比
                                        #较久的连接
        stats refresh 30                #统计页面刷新间隔
        retries 3                       #3 次连接失败就认为服务不可用,也可以通过后面设置
        balance roundrobin              #默认的负载均衡的方式,轮询方式
       #balance source                  #默认的负载均衡的方式,类似 nginx 的 ip_hash
       #balance leastconn               #默认的负载均衡的方式,最小连接
        contimeout 5000                 #连接超时
        clitimeout 50000                #客户端超时
        srvtimeout 50000                #服务器超时
        timeout check 2000              #心跳检测超时
##############监控页面的设置##########
```

```
listen admin_status                 #Frontend 和 Backend 的组合体,监控组的名称,按需自
                                    #定义名称
        bind 0.0.0.0:65532          #监听端口
        mode http                   #http 的 7 层模式
        log 127.0.0.1 local3 err    #错误日志记录
        stats refresh 5s            #每隔 5s 自动刷新监控页面
        stats uri /admin?stats      #监控页面的 url
        stats realm jfedu\ jfedu    #监控页面的提示信息
        stats auth admin:admin      #监控页面的用户和密码 admin,可以设置多个用户名
        stats hide-version          #隐藏统计页面上的 Haproxy 版本信息
        stats admin if TRUE         #手工启用/禁用,后端服务器
##########监控 haproxy 后端服务器的状态############
listen site_status
       bind 0.0.0.0:1081            #监听端口
       mode http                    #http 的 7 层模式
       log 127.0.0.1 local3 err     #[err warning info debug]
       monitor-uri /site_status     #网站健康检测 URL,用来检测 Haproxy 管理的网站是否
                                    #可以用,正常返回 200,不正常返回 503
       acl site_dead nbsrv(server_web) lt 2  #定义网站 down 时的策略当挂在负载均衡上的指
                                             #定 backend 的有效机器数小于 1 台时返回 true
       monitor fail if site_dead    #当满足策略的时候返回 503,网上文档说的是 500,实际
                                    #测试为 503
       monitor-net 192.168.149.129/32 #来自 192.168.149.129 的日志信息不会被记录和转发
       monitor-net 192.168.149.130/32 #来自 192.168.149.130 的日志信息不会被记录和转发
########frontend 配置############
#####注意,frontend 配置里面可以定义多个 acl 进行匹配操作########
frontend http_80_in
       bind 0.0.0.0:80              #监听端口,即 haproxy 提供 web 服务的端口,和 lvs 的
                                    #vip 端口类似
       mode http                    #http 的 7 层模式
       log global                   #应用全局的日志配置
       option httplog               #启用 http 的 log
       option httpclose             #每次请求完毕后主动关闭 http 通道,HA-Proxy 不支持
                                    #keep-alive 模式
       option forwardfor            #如果后端服务器需要获得客户端的真实 IP 需要配置次
                                    #参数,将可以从 Http Header 中获得客户端 IP
       ########acl 策略配置#############
       acl jfedu_web hdr_reg(host) -i ^(www1.jfedu.net|www2.jfedu.net)$
       #如果请求的域名满足正则表达式中的 2 个域名返回 true -i 是忽略大小写
       #如果请求的域名满足 www.jfedu.net 返回 true -i 是忽略大小写
       #acl jfedu hdr(host) -i jfedu.net
       #如果请求的域名满足 jfedu.net 返回 true -i 是忽略大小写
       #acl file_req url_sub -i killall=
       #在请求 url 中包含 killall=,则此控制策略返回 true,否则为 false
       #acl dir_req url_dir -i allow
```

```
        #在请求 url 中存在 allow 作为部分地址路径,则此控制策略返回 true,否则返回 false
        #acl missing_cl hdr_cnt(Content-length) eq 0
        #当请求的 header 中 Content-length 等于 0 时返回 true
########acl 策略匹配相应#############
        #block if missing_cl
        #当请求中 header 中 Content-length 等于 0 阻止请求返回 403
        #block if !file_req || dir_req
        #block 表示阻止请求,返回 403 错误,当前表示如果不满足策略 file_req,或者满足策略
        #dir_req,则阻止请求
        use_backend server_web if jfedu_web
        #当满足 jfedu_web 的策略时使用 server_web 的 backend
##########backend 的设置##############
backend server_web
        mode http                      #http 的 7 层模式
        balance roundrobin             #负载均衡的方式,roundrobin 平均方式
        cookie SERVERID              #允许插入 serverid 到 cookie 中,serverid 后面可以定义
        option httpchk GET /index.html  #心跳检测的文件
        server web1 192.168.149.129:80 cookie web1 check inter 1500 rise 3 fall 3 weight 1
        #服务器定义,cookie 1 表示 serverid 为 web1,check inter 1500 是检测心跳频率 rise 3 是 3
        #次正确认为服务器可用
        #fall 3 是 3 次失败认为服务器不可用,weight 代表权重
        server web2 192.168.149.130:80 cookie web2 check inter 1500 rise 3 fall 3 weight 2
        #服务器定义,cookie 1 表示 serverid 为 web2,check inter 1500 是检测心跳频率 rise 3 是 3
        #次正确认为服务器可用
        #fall 3 是 3 次失败认为服务器不可用,weight 代表权重
```

23.8.4 安装 keepalived 服务

```
cd /usr/src ;
wget http://www.keepalived.org/software/keepalived-1.2.1.tar.gz
tar xzf keepalived-1.2.1.tar.gz
cd keepalived-1.2.1 &&
./configure --with-kernel-dir=/usr/src/kernels/2.6.32-71.el6.x86_64/
make &&make install
DIR=/usr/local/ ;cp $DIR/etc/rc.d/init.d/keepalived  /etc/rc.d/init.d/
cp $DIR/etc/sysconfig/keepalived /etc/sysconfig/
mkdir -p /etc/keepalived && cp $DIR/sbin/keepalived /usr/sbin/
```

23.8.5 配置 Haproxy+keepalived

Haproxy+keealived master 端 keepalived.conf 配置文件如下：

```
! Configuration File for keepalived
 global_defs {
 notification_email {
```

```
        wgkgood@139.com
    }
      notification_email_from wgkgood@139.com
      smtp_server 127.0.0.1
      smtp_connect_timeout 30
      router_id LVS_DEVEL
    }
    vrrp_script chk_haproxy {
      script "/data/sh/check_haproxy.sh"
     interval 2
     weight 2
    }
    # VIP1
    vrrp_instance VI_1 {
        state MASTER
        interface eth0
        lvs_sync_daemon_inteface eth0
        virtual_router_id 151
        priority 100
        advert_int 5
        nopreempt
        authentication {
            auth_type PASS
            auth_pass 2222
        }
        virtual_ipaddress {
            192.168.0.133
        }
        track_script {
        chk_haproxy
       }
    }
```

23.8.6 创建 Haproxy 脚本

设置可执行权限 chmod +x check_haproxy.sh,脚本内容如下：

```
#!/bin/bash
#auto check haprox process
#2017-6-12 jfedu.net
killall -0 haproxy
if
   [[ $? -ne 0 ]];then
   /etc/init.d/keepalived stop
fi
```

Haproxy+keealived backup 端 keepalived. conf 配置文件如下：

```
! Configuration File for keepalived
 global_defs {
 notification_email {
      wgkgood@139.com
 }
    notification_email_from wgkgood@139.com
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
 }
 vrrp_script chk_haproxy {
    script "/data/sh/check_haproxy.sh"
   interval 2
   weight 2
 }
 # VIP1
 vrrp_instance VI_1 {
     state BACKUP
     interface eth0
     lvs_sync_daemon_interface eth0
     virtual_router_id 151
     priority 90
     advert_int 5
     nopreempt
     authentication {
         auth_type PASS
         auth_pass 2222
     }
     virtual_ipaddress {
         192.168.0.133
     }
     track_script {
     chk_haproxy
    }
 }
```

23.8.7 测试 Haproxy+keepalived 服务

手动 kill 掉 131 的 Haproxy 进程后，130 的 keepalived 后台日志显示如下，并且访问 133 VIP 正常访问，则证明 Haproxy+keepalived 高可用架构配置完毕，如图 23-4 所示。

www.jf1.com Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

(a) Haproxy+keepalived网站架构(1)

www.jf2.com Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

(b) Haproxy+keepalived网站架构(2)

```
:29:54 192-168-0-130 Keepalived_vrrp: Opening file '/etc/keepalived/keepalived.conf'

:29:54 192-168-0-130 Keepalived_vrrp: Configuration is using : 38124 Bytes
:29:54 192-168-0-130 Keepalived_vrrp: Using LinkWatch kernel netlink reflector...
:29:54 192-168-0-130 Keepalived_vrrp: VRRP_Instance(VI_1) Entering BACKUP STATE
:29:54 192-168-0-130 Keepalived_vrrp: VRRP sockpool: [ifindex(2), proto(112), fd(10,

:29:54 192-168-0-130 Keepalived_vrrp: VRRP_Script(chk_haproxy) succeeded
:31:31 192-168-0-130 Keepalived_vrrp: VRRP_Instance(VI_1) Transition to MASTER STATE
:31:36 192-168-0-130 Keepalived_vrrp: VRRP_Instance(VI_1) Entering MASTER STATE
:31:36 192-168-0-130 Keepalived_vrrp: VRRP_Instance(VI_1) setting protocol VIPs.
:31:36 192-168-0-130 Keepalived_vrrp: VRRP_Instance(VI_1) Sending gratuitous ARPs on
192.168.0.133
:31:41 192-168-0-130 Keepalived_vrrp: VRRP_Instance(VI_1) Sending gratuitous ARPs on
192.168.0.133
```

(c) Haproxy+keepalived网站架构(3)

HAProxy

Statistics Report for pid 7658

> General process information

pid = 7658 (process #1, nbproc = 1)
uptime = 0d 0h03m31s
system limits: memmax = unlimited; ulimit-n = 8204
maxsock = 8204; **maxconn** = 4096; **maxpipes** = 0
current conns = 1; current pipes = 0/0
Running tasks: 1/1

active UP | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
Note: UP with load-balancing disabled is reported as "NOLB".

Display option:
- Hide 'DO
- Disable re
- Refresh n
- CSV expo

http-in

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings					
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act
Frontend				0	45	-	0	3	2 000	147		49 991	38 544	0	0	36					OPEN			

www.wugk1.com

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings					Se
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act
Server1	0	0	-	0	25		0	1	-	58	0	27 041	16 770		0		0	0	0	0			1	Y
Backend	0	0		0	25		0	1	0	58	0	27 041	16 770	0	0		0	0	0	0	3m31s UP		1	1

www.wugk2.com

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings					Se
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act
Server1	0	0	-	0	46		0	2	-	53	0	22 950	21 587		0		0	0	0	0			1	Y
Backend	0	0		0	46		0	2	0	53	0	22 950	21 587	0	0		0	0	0	0	3m31s UP		1	1

admin_stats

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings					S
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Ac
Frontend				1	4	-	1	2	2 000	24		23 171	122 724	0	0	0					OPEN			
Backend	0	0		0	2		0	1	2 000	13	0	23 171	122 724	0	0		13	0	0	0	3m31s UP		0	0

(d) Haproxy+keepalived网站架构(4)

图 23-4 Haproxy+keepalived 网站架构

23.9 LVS＋keepalived 高可用集群实战

LVS 是 Linux virtual server 的简写，意即 Linux 虚拟服务器，是一个虚拟的服务器集群系统。本项目在 1998 年 5 月由章文嵩博士成立，是中国国内最早出现的自由软件项目之一。LVS 简单工作原理为用户请求 LVS VIP，LVS 根据转发方式和算法，将请求转发给后端服务器，后端服务器接收到请求，返回给用户。对于用户来说，看不到 Web 后端具体的应用。

23.9.1 LVS 负载均衡简介

互联网主流可伸缩网络服务有很多结构，但是都有一个共同点，它们都需要一个前端的负载调度器（或者多个进行主从备份）。实现虚拟网络服务的主要技术指出 IP 负载均衡技术是在负载调度器的实现技术中效率最高的。

已有的 IP 负载均衡技术中，主要有通过网络地址转换（network address translation）将一组服务器构成一个高性能的、高可用的虚拟服务器，通常称之为 VS/NAT 技术（virtual server via network address translation）。

在分析 VS/NAT 的缺点和网络服务的非对称性的基础上，可以通过 IP 隧道实现虚拟服务器的方法 VS/TUN（virtual server via IP tunneling）和通过直接路由实现虚拟服务器的方法 VS/DR（virtual server via direct routing），它们可以极大地提高系统的伸缩性。

总体来说，IP 负载均衡技术分为 VS/NAT、VS/TUN 和 VS/DR 技术，它们是 LVS 集群中实现的三种 IP 负载均衡技术。

23.9.2 LVS 负载均衡工作原理

实现 LVS 负载均衡转发方式有三种，分别为 NAT、DR、TUN 模式，LVS 均衡常见算法包括 RR（round-robin）、LC（least_connection）、W（weight）RR、WLC 模式等（RR 为轮询模式，LC 为最少连接模式）。

LVS NAT 原理：用户请求 LVS 到达 director，director 将请求的报文的目标 IP 地址改成后端的 realserver IP 地址，同时将报文的目标端口也改成后端选定的 realserver 相应端口，最后将报文发送到 realserver，realserver 将数据返给 director，director 再把数据发送给用户。因为两次请求都经过 director，所以访问量大的话，director 会成为瓶颈，如图 23-5 所示。

LVS DR 原理：用户请求 LVS 到达 director，director 将请求的报文的目标 MAC 地址改成后端的 realserver MAC 地址，目标 IP 为 VIP（不变），源 IP 为用户 IP 地址（保持不变），然后 director 将报文发送到 realserver，realserver 检测到目标为自己本地 VIP，如果在同一个网段，然后将请求直接返给用户。如果用户跟 realserver 不在一个网段，则通过网关返回用户，如图 23-6 所示。

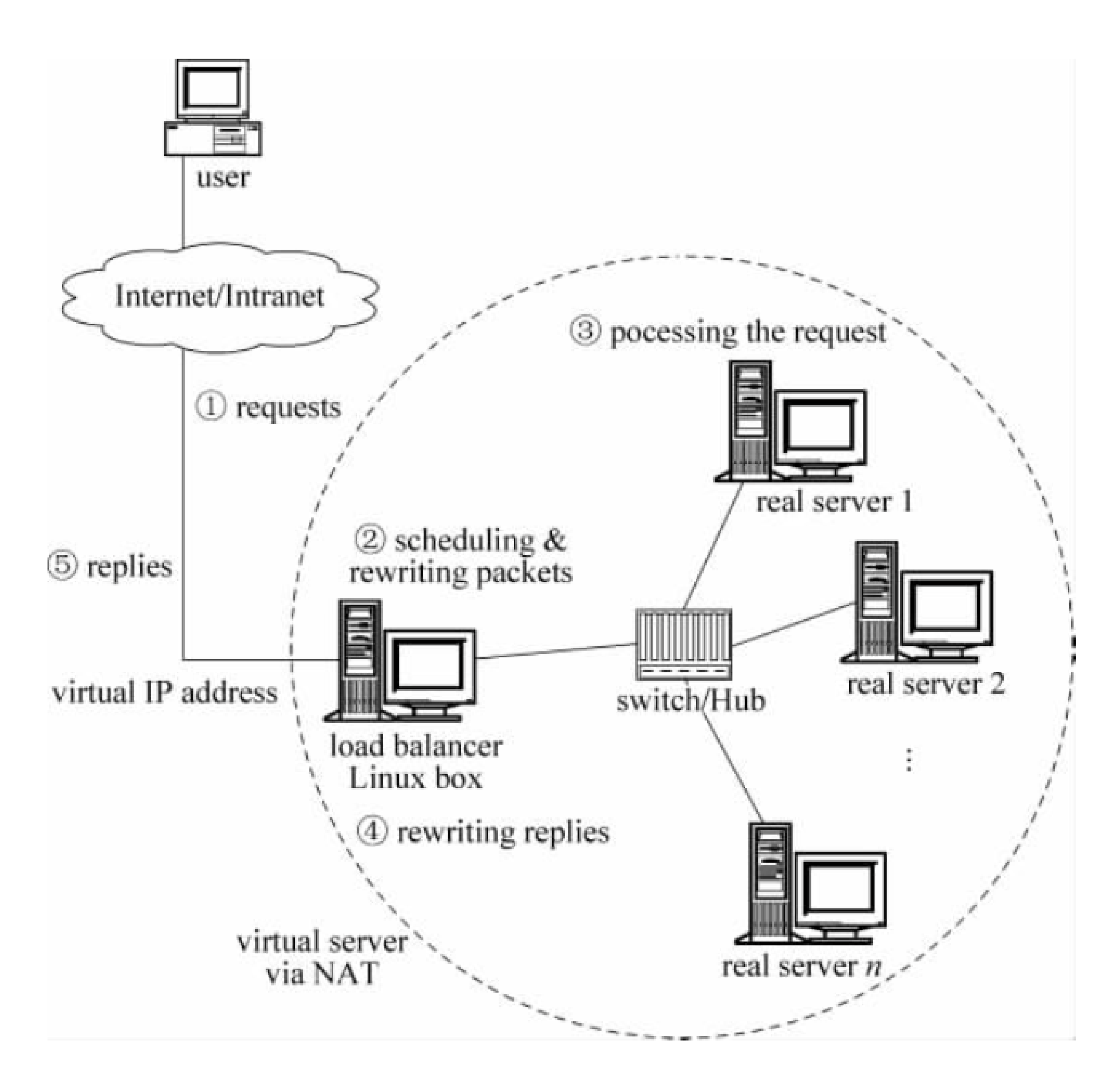


图 23-5　LVS NAT 原理详解图

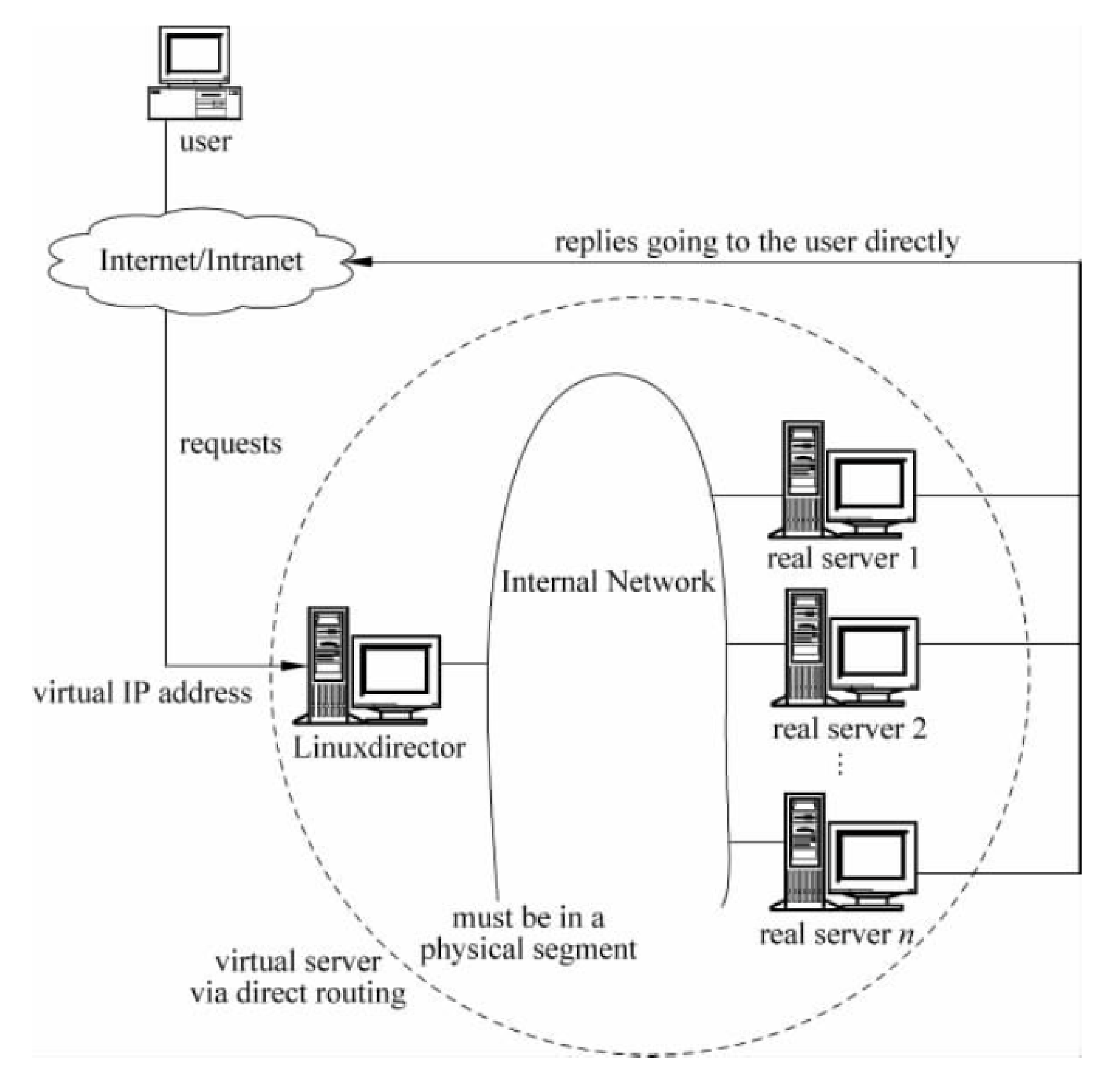


图 23-6　LVS DR 原理详解图

LVS TUN 原理：用户请求 LVS 到达 director，director 通过 IP-TUN 加密技术将请求的报文的目标 MAC 地址改成后端的 realserver MAC 地址，目标 IP 为 VIP（不变），源 IP 为用户 IP 地址（保持不变），然后 director 将报文发送到 realserver，realserver 基于 IP-TUN 解密，然后检测到目标为自己本地 VIP，如果在同一个网段，然后将请求直接返给用户。如果用户跟 realserver 不在一个网段，则通过网关返回用户，如图 23-7 所示。

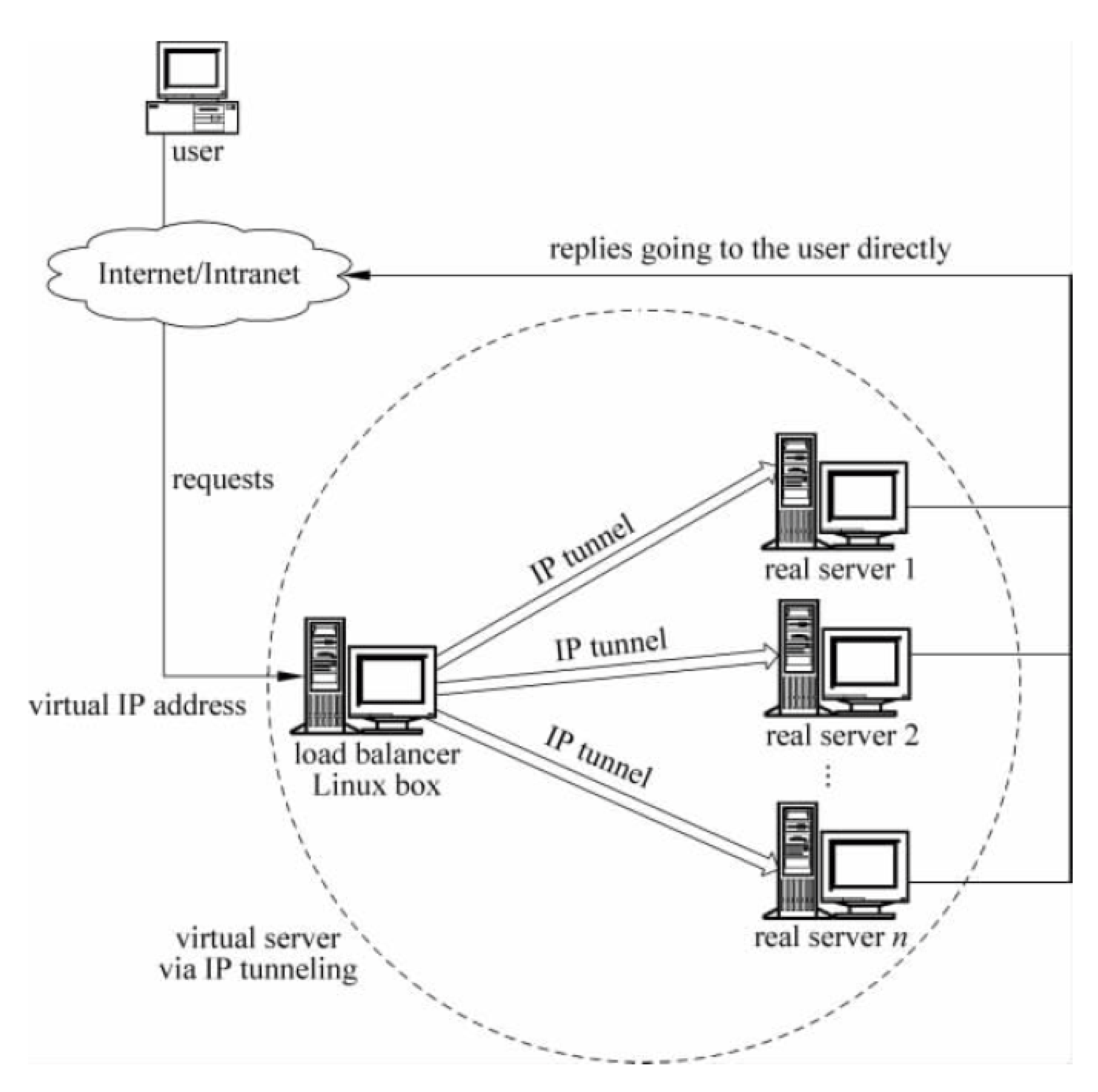


图 23-7　LVS TUN 原理详解图

23.9.3　LVS 负载均衡实战配置

LVS 负载均衡技术实现是基于 Linux 内核模块 ipvs，与 iptables 一样是直接工作在内核中，互联网主流的 Linux 发行版默认都已经集成了 ipvs 模块，因此只需安装管理工具 ipvsadm，所需软件为 ipvsadm-1.2.4.tar.gz 软件，安装配置步骤如下：

（1）ipvsadm 编译安装方法如下：

```
wget -c
http://www.linuxvirtualserver.org/software/kernel-2.6/ipvsadm-1.24.tar.gz
ln -s /usr/src/kernels/2.6.* /usr/src/linux
tar xzvf ipvsadm-1.24.tar.gz
cd ipvsadm-1.24
make
make install
```

（2）ipvsadm 软件安装完毕后，需要进行配置，主要配置方法有三步：添加虚拟服务器IP、添加 realserver 后端服务及启动 LVS 服务器 VIP 地址，配置代码如下：

```
ipvsadm -A -t 192.168.33.188:80 -s rr
ipvsadm -a -t 192.168.33.188:80 -r 192.168.33.12 -g -w 2
ipvsadm -a -t 192.168.33.188:80 -r 192.168.33.13 -g -w 2
```

（3）可以使用 shell 脚本自动部署 LVS 相关软件及配置，代码如下：

```
#!/bin/bash

SNS_VIP= $2
SNS_RIP1= $3
SNS_RIP2= $4
if [ "$1" == "stop" -a -z "$2" ];then
    echo "------------------------------------------------"
    echo -e "\033[32mPlease Enter $0 stop LVS_VIP\n\nEXample: $0 stop 192.168.1.111\033[0m"
    echo
    exit
else
    if [ -z "$2" -a -z "$3" -a -z "$4" ];then
        echo "----------------------------------------------"
         echo -e "\033[32mPlease Enter Input $0 start VIP REALSERVER1 REALSERVER2\n\
nEXample: $0 start/stop 192.168.1.111 192.168.1.2 192.168.1.3\033[0m"
        echo
        exit 0
    fi
fi
. /etc/rc.d/init.d/functions
logger $0 called with $1
function IPVSADM(){
/sbin/ipvsadm --set 30 5 60
/sbin/ifconfig eth0:0 $SNS_VIP broadcast $SNS_VIP netmask 255.255.255.255 broadcast $SNS_
VIP up
/sbin/route add -host $SNS_VIP dev eth0:0
/sbin/ipvsadm -A -t $SNS_VIP:80 -s wlc -p 120
/sbin/ipvsadm -a -t $SNS_VIP:80 -r $SNS_RIP1:80 -g -w 1
/sbin/ipvsadm -a -t $SNS_VIP:80 -r $SNS_RIP2:80 -g -w 1
}
case "$1" in
start)
IPVSADM
echo "------------------------------------------------------------"
/sbin/ipvsadm -Ln
touch /var/lock/subsys/ipvsadm > /dev/null 2 > &1
;;
stop)
/sbin/ipvsadm -C
/sbin/ipvsadm -Z
```

```
ifconfig eth0:0 down >>/dev/null 2>&1
route del  $SNS_VIP >>/dev/null 2>&1
rm -rf /var/lock/subsys/ipvsadm > /dev/null 2>&1
echo "ipvsadm stopped!"
;;
status)
if [ ! -e /var/lock/subsys/ipvsadm ]
then
echo "ipvsadm stopped!"
exit 1
else
echo "ipvsadm started!"
fi
;;
*)
echo "Usage: $0 {start | stop | status}"
exit 1
esac
exit 0
```

（4）LVS 服务器绑定 VIP 地址，命令如下：

```
VIP=192.168.33.188
ifconfigeth0:0 $VIP netmask 255.255.255.255 broadcast $VIP
/sbin/route add-host $VIP dev eth0:0
```

（5）LVS ipvsadm 配置参数说明如下：

- -A：增加一台虚拟服务器 VIP 地址。
- -t：虚拟服务器提供的是 TCP 服务。
- -s：使用的调度算法。
- -a：在虚拟服务器中增加一台后端真实服务器。
- -r：指定真实服务器地址。
- -w：后端真实服务器的权重。
- -m：设置当前转发方式为 NAT 模式。
- -g：模式为直接路由模式。
- -i：模式为隧道模式。

（6）查看 LVS 转发列表命令为 ipvsadm -Ln，如图 23-8 所示。

```
[root@node2 ~]#
[root@node2 ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.149.129:80 rr
  -> 192.168.149.131:80           Masq    2      0          0
  -> 192.168.149.130:80           Masq    2      0          0
[root@node2 ~]#
```

图 23-8 LVS ipvsadm 查看链接状态

（7）Nginx 客户端 realserver 配置 VIP 脚本如下：

```
#!/bin/sh
#LVS Client Server
VIP=192.168.33.188
case $1 in
start)
    ifconfig lo:0 $VIP netmask 255.255.255.255 broadcast $VIP
    /sbin/route add -host $VIP dev lo:0
    echo "1" >/proc/sys/net/ipv4/conf/lo/arp_ignore
    echo "2" >/proc/sys/net/ipv4/conf/lo/arp_announce
    echo "1" >/proc/sys/net/ipv4/conf/all/arp_ignore
    echo "2" >/proc/sys/net/ipv4/conf/all/arp_announce
    sysctl -p >/dev/null 2>&1
    echo "RealServer Start OK"
    exit 0
;;
stop)
    ifconfig lo:0 down
    route del $VIP >/dev/null 2>&1
    echo "0" >/proc/sys/net/ipv4/conf/lo/arp_ignore
    echo "0" >/proc/sys/net/ipv4/conf/lo/arp_announce
    echo "0" >/proc/sys/net/ipv4/conf/all/arp_ignore
    echo "0" >/proc/sys/net/ipv4/conf/all/arp_announce
    echo "RealServer Stoped OK"
    exit 1
;;
*)
    echo "Usage: $0 {start|stop}"
;;
esac
```

如果单台 LVS 发生突发情况，例如宕机、发生不可恢复，会导致用户无法访问后端所有的应用程序。为了避免这种问题导致网站长时间不可用，可以使用 HA 方案实现故障切换，可以基于 LVS+keepalived 实现负载均衡及高可用功能，满足网站 7×24 小时稳定高效地运行。

keepalived 基于三层检测（IP 层、TCP 层、应用层），主要用于检测 Web 服务器的状态，如果有一台 Web 服务器死机，或工作出现故障，keepalived 检测到并将有故障的 Web 服务器从系统中剔除。

当后端一台 Web 服务器工作正常后 keepalived 自动将 Web 服务器加入到服务器群中，这些工作全部自动完成，不需要人工干涉，需要人工做的只是修复故障的 Web 服务器。

需要注意，如果使用了 keepalived.conf 配置，就不需要再执行 ipvsadm -A 命令去添加均衡的 realserver 命令了，所有的配置都在 keepalived.conf 里面设置即可。

23.9.4 LVS+keepalived 实战配置

LVS+keepalived 负载均衡高可用集群架构适用于千万级并发网站，在互联网企业得到广泛的应用，以下为 LVS+keepalived 企业级配置方法和步骤。

（1）ipvsadm 编译安装方法如下：

```
wget -c
http://www.linuxvirtualserver.org/software/kernel-2.6/ipvsadm-1.24.tar.gz
ln -s /usr/src/kernels/2.6.* /usr/src/linux
tar xzvf ipvsadm-1.24.tar.gz
cd ipvsadm-1.24
make
make install
```

（2）keepalived 安装配置如下：

```
cd /usr/src
wget -c http://www.keepalived.org/software/keepalived-1.1.15.tar.gz
tar -xzvf keepalived-1.1.15.tar.gz
cd keepalived-1.1.15
./configure
make && make install
DIR=/usr/local/
cp $DIR/etc/rc.d/init.d/keepalived /etc/rc.d/init.d/
cp $DIR/etc/sysconfig/keepalived /etc/sysconfig/
mkdir -p /etc/keepalived
cp $DIR/sbin/keepalived /usr/sbin/
```

（3）master 上 keepalived.conf 配置代码如下：

```
! Configuration File for keepalived
global_defs {
   notification_email {
      wgkgood@163.com
   }
   notification_email_from wgkgood@163.com
   smtp_server 127.0.0.1
   smtp_connect_timeout 30
   router_id LVS_DEVEL
}
# VIP1
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    lvs_sync_daemon_inteface eth0
    virtual_router_id 51
    priority 100
```

```
    advert_int 5
    nopreempt
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.33.188
    }
}
virtual_server 192.168.33.188 80 {
    delay_loop 6
    lb_algo wrr
    lb_kind DR
    persistence_timeout 60
      protocol TCP
        real_server 192.168.33.12 80 {
        weight 100
        TCP_CHECK {
        connect_timeout 10
        nb_get_retry 3
        delay_before_retry 3
        connect_port 80
        }
    }
    real_server 192.168.33.13 80 {
        weight 100
        TCP_CHECK {
        connect_timeout 10
        nb_get_retry 3
        delay_before_retry 3
        connect_port 80
        }
    }
}
```

（4）backup 上 keepalived.conf 配置代码如下：

```
! Configuration File for keepalived
global_defs {
   notification_email {
      wgkgood@163.com
   }
   notification_email_from wgkgood@163.com
   smtp_server 127.0.0.1
   smtp_connect_timeout 30
```

```
    router_id LVS_DEVEL
}
# VIP1
vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    lvs_sync_daemon_inteface eth0
    virtual_router_id 51
    priority 90
    advert_int 5
    nopreempt
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.33.188
    }
}
virtual_server 192.168.33.188 80 {
    delay_loop 6
    lb_algo wrr
    lb_kind DR
    persistence_timeout 60
    protocol TCP
    real_server 192.168.33.12 80 {
        weight 100
        TCP_CHECK {
        connect_timeout 10
        nb_get_retry 3
        delay_before_retry 3
        connect_port 80
        }
    }
    real_server 192.168.33.13 80 {
        weight 100
        TCP_CHECK {
        connect_timeout 10
        nb_get_retry 3
        delay_before_retry 3
        connect_port 80
        }
    }
}
```

Master keepalived 配置 state 状态为 MASTER，priority 设置为 100，backup keepalived 配置 state 状态为 BACKUP，priority 设置 90，转发方式为 DR 直连路由模式，算采用 wrr 模

式，在 LVS backup 服务器写入如上配置，需要注意客户端的配置要修改优先级及状态。

LVS+keepalived 负载均衡主备配置完毕，由于 LVS 采用 DR 模式，根据 DR 模式转发原理，需在客户端 realserver 绑定 VIP。

23.9.5 LVS DR 客户端配置 VIP

用户通过浏览器访问 director 的 VIP，director 接收请求，将通过相应的转发方式及算法将请求转发给相应的 realserver。在转发的过程中，会修改请求包的目的 MAC 地址，目的 IP 地址不变。realserver 接收请求，并直接响应客户端。

director 与 realserver 位于同一个物理网络中，当 director 直接将请求转发给 realserver 时，如果 realserver 检测到该请求包目的 IP 是 VIP 而并非自己，便会丢弃，不会响应。

为了解决这个问题，需在所有 realserver 上都配上 VIP，保证数据包不丢弃，同时由于后端 realserver 都配置 VIP 会导致 IP 冲突，所以需将 VIP 配置在 lo 网卡上，这样限制了 VIP 不会在物理交换机上产生 MAC 地址表，从而避免 IP 冲突。LVS DR 客户端启动 realserver.sh 脚本内容如下：

```
#!/bin/sh
#LVS Client Server
VIP=192.168.33.188
case $1 in
start)
    ifconfig lo:0 $VIP netmask 255.255.255.255 broadcast $VIP
    /sbin/route add -host $VIP dev lo:0
    echo "1" >/proc/sys/net/ipv4/conf/lo/arp_ignore
    echo "2" >/proc/sys/net/ipv4/conf/lo/arp_announce
    echo "1" >/proc/sys/net/ipv4/conf/all/arp_ignore
    echo "2" >/proc/sys/net/ipv4/conf/all/arp_announce
    sysctl -p >/dev/null 2>&1
    echo "RealServer Start OK"
    exit 0
;;
stop)
    ifconfig lo:0 down
    route del $VIP >/dev/null 2>&1
    echo "0" >/proc/sys/net/ipv4/conf/lo/arp_ignore
    echo "0" >/proc/sys/net/ipv4/conf/lo/arp_announce
    echo "0" >/proc/sys/net/ipv4/conf/all/arp_ignore
    echo "0" >/proc/sys/net/ipv4/conf/all/arp_announce
    echo "RealServer Stoped OK"
    exit 1
;;
*)
    echo "Usage: $0 {start|stop}"
;;
```

```
esac
```

23.9.6 LVS 负载均衡企业实战排错经验

LVS 在企业生产环境中如何去排错呢，遇到问题该怎么办呢？LVS 使用过程中，会遇到很多问题，以下为 LVS 故障排错思路。

企业网站 LVS+keepalived+Nginx 架构中，突然发现网站 www.jfedu.net 部分用户访问巨慢，甚至无法访问，这个问题该如何定位呢？思路如下：

(1) 客户端 ping www.jfedu.net，通过 ping 返回域名对应的 IP 是否正常；

(2) 如果无法返回 IP，或者响应比较慢，定位 DNS 或者网络延迟问题，可以通过 tracert www.jfedu.net 测试客户端本机到服务器的链路延迟；

(3) 登录 LVS 服务器，ipvsadm -Ln 查看当前后端 Web 连接信息，显示如下：

```
[root@LVS-Master keepalived]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
 -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP 192.168.1.10:80 wlc
 -> 192.168.1.6:80               Route   100    2          13
 -> 192.168.1.5:80               Route   100    120        13
 -> 192.168.1.4:80               Route   100    1363       45
```

通过 LVS ipvsadm 信息，看到 LVS 选择的轮训方式为加权最少连接，而网站也是部分无法访问，猜测是其中一台 Web 服务器无法访问或者访问巨慢导致，如果单台 Web 异常，为什么 LVS+keepalived 不能将异常的 Web 服务器 IP 异常均衡列表呢？

查看 keepalived.conf 负载均衡健康检查配置，部分代码如下：

```
real_server 192.168.1.4 80 {
        weight 100
        TCP_CHECK {
        connect_timeout 10
        nb_get_retry 3
        delay_before_retry 3
        connect_port 80
    }
}
```

通过配置文件发现 LVS 默认用的是 TCP 检测方式，只要 80 端口能通，请求就会被转发到后端服务器。紧接着在 LVS 服务器使用 wget http://192.168.1.4/返回很慢，直至超时，而另外几台 Nginx realserver 返回正常，查看 Nginx 192.168.1.4 服务器 80 端口服务正常启动，所以对于 LVS 服务器来说是打开的，所以 LVS 会把请求转发给它。

为什么部分用户可以访问，有的用户无法访问呢？发现 192.168.1.4 服务器 ifconfig 查看 IP，但是没有看到 VIP 地址绑定在 lo:0 网卡上，经排错由于该服务器被重启，

realserver 脚本配置 VIP 异常，启动 realserver 脚本，网站恢复正常。

为了防止上述突发问题，增加 LVS 对后端 Nginx URL 的检测，能访问 URL 则表示服务正常。对比之前的检测方式，从单纯的 80 端口到现在的 URL 检测，后端如果某台出现 502 超时错误，keepalived 列表会自动踢出异常的 realserver，等待后端 realserver 恢复后自动添加到服务器正常列表。keepalived 基于 URL 检查代码如下：

```
real_server 192.168.1.4 80 {
        weight 100
        HTTP_GET {
        url {
        path /monitor/warn.jsp
        status_code 200
         }
        connect_timeout 10
        nb_get_retry 3
        delay_before_retry 3
     }
}
```

第 24 章



实战 Docker 虚拟化技术

Docker 虚拟化技术是目前最火爆的技术之一,Docker 旨在实现应用软件、程序快速打包部署及容器快速部署,为生产环境更快集成及交付提供便利。

本章向读者介绍 Docker 虚拟化技术入门、Docker 简介、Docker 镜像、Docker 容器、Docker 仓库及 Docker 安装使用、DockerFile 构建镜像、Docker 磁盘管理、扩容及一键构建 Docker 虚拟化平台等内容。

24.1 虚拟化概述及简介

1961 年 IBM709 机实现了分时系统,将 CPU 占用切分为多个极短(1/100s)时间片,每一个时间片都执行着不同的任务。通过对这些时间片的轮询,就可以将一个 CPU 虚拟化或者伪装成为多个 CPU,并且让每一个虚拟 CPU 看起来都是在同时运行,这就是虚拟机的雏形。

1972 年 IBM 正式将 system370 机的分时系统命名为虚拟机,1990 年 IBM 推出的 system390 机支持逻辑分区,将一个 CPU 分为若干份(最多 10 份),而且每份 CPU 都是独立的,即一个物理 CPU 可以在逻辑上分为 10 个 CPU。

直到 IBM 将分时系统开源后,个人 PC 终于迎来了虚拟化的开端,后来才有了 VMware、VirtualBox、Hyper-V、KVM、Xen 等虚拟化技术的发展,至今为止仍有部分软件应用分时系统作为虚拟化的基础实现。

虚拟化是一种资源管理技术,能够把物理资源转变为逻辑上可以管理的资源,如服务器、网络、内存及存储等资源,虚拟化可以打破物理结构间的壁垒,计算元件运行在虚拟的基础上而非真实的基础上,可以扩大硬件的容量,简化软件的重新配置过程,从而让用户可以用比原本的组态更好的方式来应用和管理这些资源。

虚拟化技术允许一个平台同时运行多个操作系统,并且应用程序可以在相互独立的空间内运行而互不影响,从而显著提高计算机的工作效率,它是一个为了简化管理、优化资源的解决方案。

目前企业主流的虚拟化技术包括：KVM、Xen、VMware Esxi、VirtualBox、Docker，虚拟化技术越来越广泛的应用在互联网企业中，如百度、京东、淘宝、腾讯等。

虚拟化原理：虚拟化解决方案的底部是要进行虚拟化的物理机器，该物理机器可以直接支持虚拟化，或间接支持虚拟化，那么就需要虚拟机管理程序层的支持。虚拟机管理程序(virtual machine monitor，VMM)，可以看作是平台硬件和操作系统之间的抽象化或中间件。

VMM为每个虚拟机分配一套数据结构来管理它们的状态，包括虚拟处理器的全套寄存器、物理内存的使用情况、网络设备状态、虚拟设备状态等。

VMM可以对底层(HostOS)硬件资源(物理CPU、内存、磁盘、网卡、显卡等)进行封装、隔离，抽象为另一种形式的逻辑资源，再提供给上层(GuestOS)虚拟机使用，通过虚拟化技术实现的虚拟机被称为GuestOS，而作为GuestOS载体的物理主机称为HostOS(宿主)。

虚拟化的分层抽象如图24-1所示。

虚拟化技术在技术实现上通常分为完全虚拟化、半虚拟化、轻量级虚拟化，以下为这三种虚拟化技术实现的区别：

- 完全拟化技术，实际上是通过软件实现对操作系统的资源再分配，比较成熟，如KVM、VirtualBox等；
- 半虚拟化技术，则是通过代码修改已有的系统，形成一种新的可虚拟化的系统，调用硬件资源去安装多个系统，整体速度上相对高一点，代表产品有Xen；
- 轻量级虚拟化，介于完全虚拟化和半虚拟化之间，典型代表有Docker。

userspace (applications) | userspace (applications)
GuestOS (virtual machine) | GuestOS (virtual machine)
hypervisor (virtual machine monitor)
hardware

图24-1　虚拟化分层抽象

24.2　Docker入门简介

Docker技术类似码头上看到的集装箱，最早集装箱没有出现的时候，码头上有许多搬运的工人在搬运货物，有了集装箱以后，搬运货物变得简单，通过集装箱的搬运模式更加单一、高效，将货物打包在集装箱里面，可以防止货物之间相互影响。

如果要将货物搬运到另外一个码头就需要转运，通过集装箱，可以直接把它们运送到另一个船舱内，而且完全可以保证里面的货物是整体地搬迁，而不会损坏货物本身。

而Docker虚拟化正是基于类似的原理，将原本复杂的环境打包成为镜像模板，然后将模板迁移到各个平台，可以快速地交付使用，从而减少了人工大量的干预。

Docker是一个开源的应用容器引擎，开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的Linux机器上，进而实现虚拟化。

容器是完全使用沙箱机制的，而且相互之间不会有任何接口，几乎没有性能开销，可以很容易地在机器和数据中心中运行，最重要的是，它们不依赖于任何语言、框架或包括系统。Docker 应该是近些年最火爆的技术之一，而且在 2018 年将开启新的跨越。

Docker 自开源后受到广泛的关注和讨论，以至于 dotCloud 公司后来都改名为 Docker Inc，直至最新改名为 Moby。Redhat 已经在其 RHEL6.5 中集中支持 Docker，Google 也在其平台即服务(platform -as -a -service，PaaS)的产品中广泛应用 Docker。

Docker 项目的目标是实现轻量级的操作系统虚拟化解决方案，Docker 的基础是 Linux 容器(Linux container，LXC)等技术，Docker 在 LXC 技术的基础上进行了进一步的封装，让用户不需要去关心容器的管理，使得操作更为简便，用户操作 Docker 的容器就像操作一个快速轻量级的虚拟机一样简单。

Docker 虚拟化和传统虚拟化(KVM、Xen 等)方式的不同之处在于 Docker 虚拟化可以在操作系统层面上直接实现 App 或者应用虚拟化，直接复用本地主机的操作系统，而传统方式则需要在硬件的基础上，虚拟 GuestOS 操作系统，然后在 GuestOS 操作系统上部署相关的 App 应用。

传统虚拟化方案分层如图 24-2 所示。

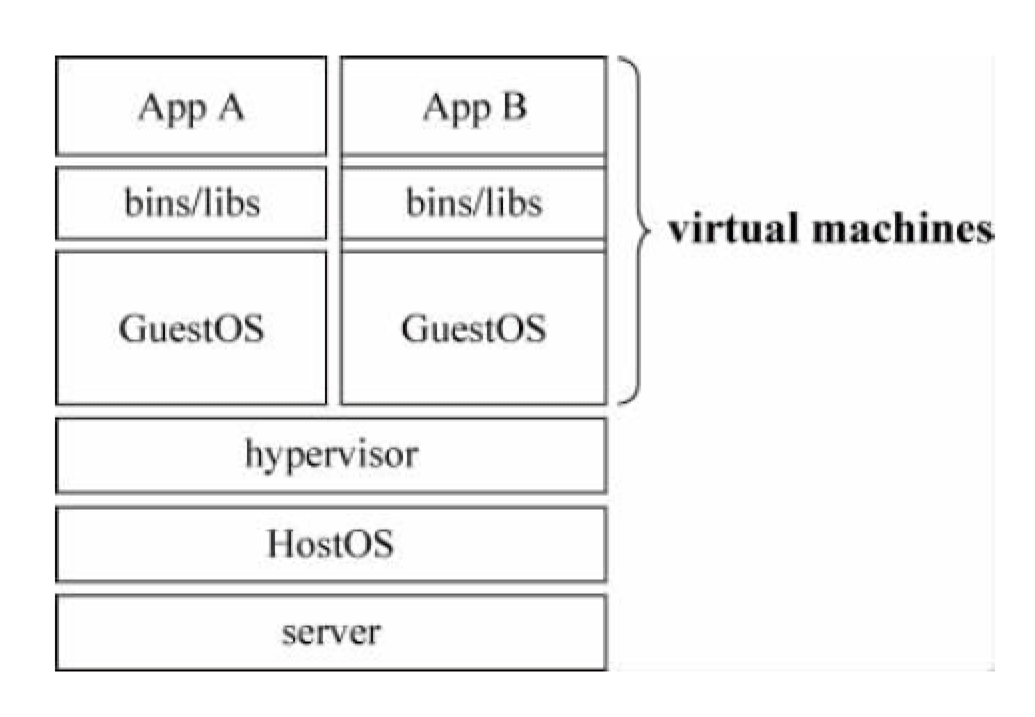


图 24-2　传统虚拟化分层抽象

Docker 虚拟化方案分层如图 24-3 所示。

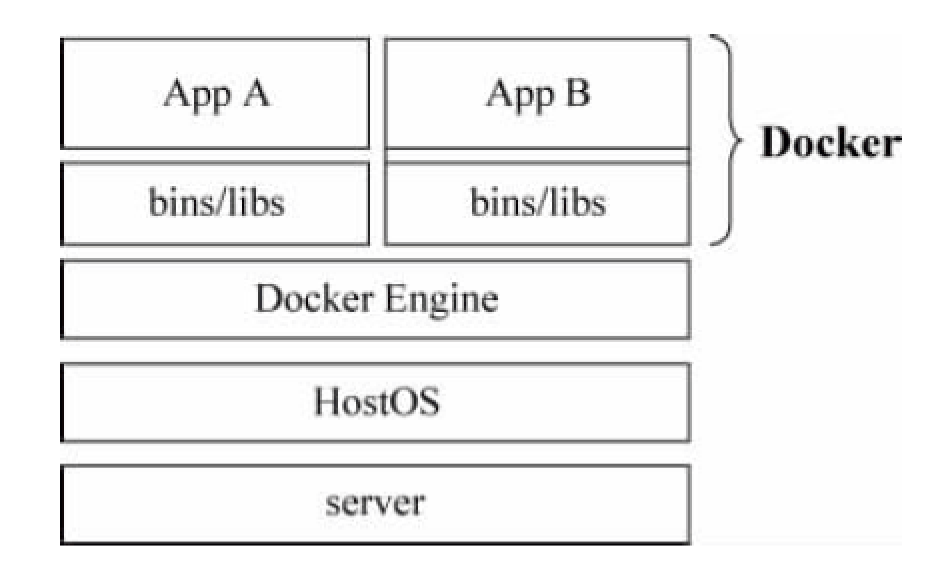


图 24-3　Docker 虚拟化分层抽象

通过如上对比，可以得出 Docker 虚拟化可以直接在 HostOS 上基于 VMM 启动各种 App，而传统虚拟化需要在 VMM 上启动 GuestOS，然后在 GuestOS 上再启动 App。

Docker 虚拟化实施有以下三个概念：

- Docker 镜像，Docker 镜像是一个静态模板，与常见的 ISO 镜像类似，是一个样板，不能直接修改，可以通过封装生成；
- Docker 容器，基于 Docker 镜像运行启动的应用或系统，称之为一个 Docker 容器或 Docker 虚拟机；
- Docker 仓库，Docker 仓库是存放 Docker 镜像的地方，常见分为公开仓库（public）和私有仓库（private）两种形式。

24.3　Docker LXC 及 Cgroup

Docker 虚拟化的由来需要从 Docker 发展历史来看，最早的 Docker 技术为 LXC＋联合文件系统（another union file system，AUFS）组合。Docker 0.9.0 版本开始引入 libcontainer，可以视作 LXC 的替代品，其中 LXC 负责资源管理，AUFS 负责镜像管理。而 LXC 包括 Cgroup（control groups）、Namespace、Chroot 等组件，并通过 Cgroups 进行资源管理。

从资源管理来看，Docker、LXC、Cgroup 三者的关系是 Cgroup 在最底层落实资源管理，LXC 在 Cgroups 上封装了一层，Docker 又在 LXC 封装了一层，要想学好 Docker，需要了解负责资源管理的 Cgroups 和 LXC。

Cgroups 是 Linux 内核提供的一种可以限制、记录、隔离进程组（process groups）所使用的物理资源，如 CPU、Memory、硬盘 IO、网卡流量等。Cgroup 最初由 Google 的工程师提出，后来被整合进 Linux 内核，Cgroups 是 LXC 为实现虚拟化所使用的资源管理手段，可以说没有 Cgroups 就没有 LXC，没有 LXC 就没有 Docker。

Cgroups 最初的目标是为资源管理提供一个统一的框架，既整合现有的 Cpuset 等子系统，也为未来开发新的子系统提供接口。现在的 Cgroups 适用于多种应用场景，从单个进程的资源控制，到实现操作系统层次的虚拟化（OS level virtualization）。

Linux container 容器可以提供轻量级的虚拟化，以便隔离进程和资源，而且不需要提供指令解释机制以及全虚拟化的其他复杂性。容器有效地将由单个操作系统管理的资源划分到孤立的组中，以便更好地在孤立的组之间平衡有冲突的资源使用需求。

传统典型的 Linux 文件系统由 bootfs 和 rootfs 两部分组成，bootfs（boot file system）主要包含 bootloader 和 kernel，bootloader 主要是引导加载 kernel，当 kernel 被加载到内存中后 bootfs 会被 umount，rootfs（root file system）包含 Linux 系统中的/dev，/proc，/bin，/etc 等目录和文件等，如图 24-4 所示。

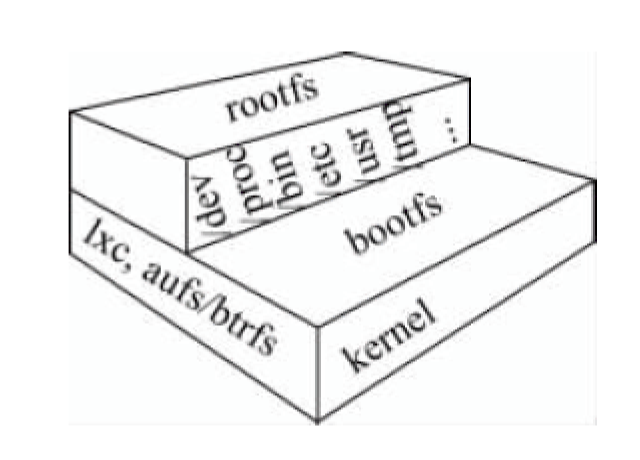


图 24-4　Linux 文件系统组成

Docker 容器的文件系统最早是 AUFS 上，AUFS 支持

将不同的目录挂载到同一个虚拟文件系统下，并实现一种分层 layer 的概念。由于 AUFS 未能加入到 Linux 内核，考虑到兼容性问题，加入了 device mapper 的支持，Docker 目前默认运行在 device mapper 文件系统基础上。

传统的 Linux 系统加载 bootfs 时会先将 rootfs 设为 read-only，系统自检之后将 rootfs 从 read-only 改为 read-write，然后用户可以在 rootfs 上进行写和读操作。但 Docker 的镜像却不是这样，它在 bootfs 自检完毕之后并不会把 rootfs 的 read-only 改为 read-write。而是利用 union mount（UnionFS 的一种挂载机制）将一个或多个 read-only 的 rootfs 加载到之前的 read-only 的 rootfs 层之上。

加载多层 rootfs 之后，整体还是一个文件系统，在 Docker 的体系里把 union mount 的这些 read-only 的 rootfs 叫作 Docker 的镜像。每一层 rootfs 都是 read-only 的，此时还不能对其进行操作，当创建一个容器，需要将 Docker 镜像进行实例化，系统会在一层或是多层 read-only 的 rootfs 之上分配一层空的可读写的 read-write 的 rootfs，所以一个完整的 Docker 镜像是由单层或多层只读镜像和一层可读可写层镜像组成，如图 24-5 所示。

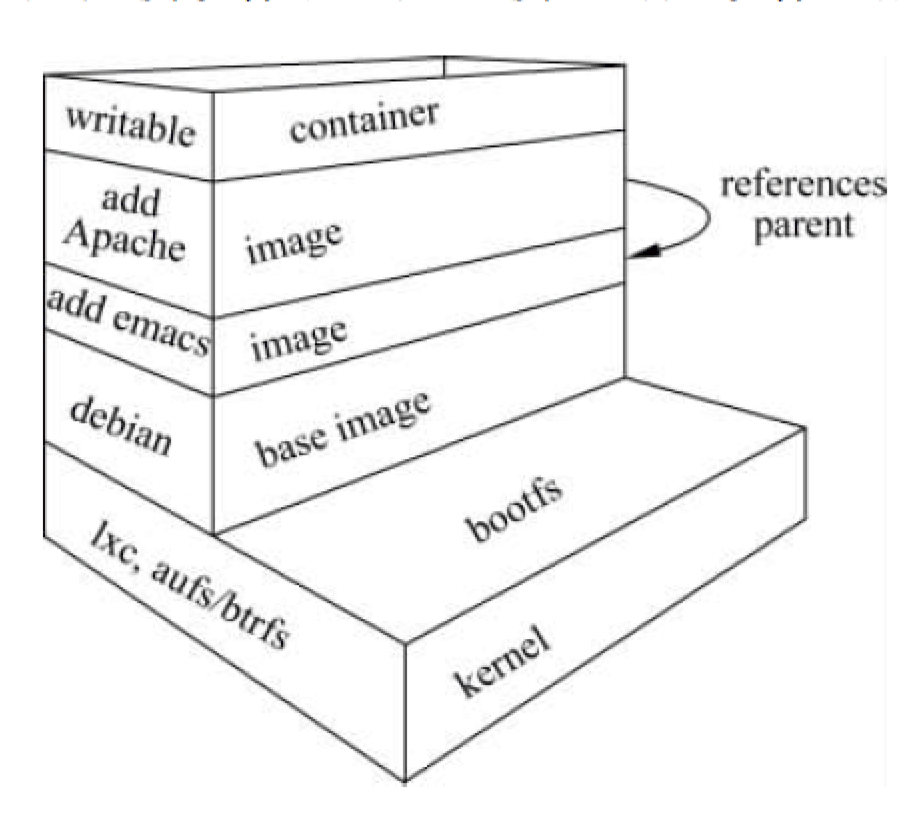


图 24-5　Docker 镜像分层

对于 Docker 另外一种文件系统 device mapper 来说，device mapper 是 Linux 2.6 内核中支持逻辑卷管理的通用设备映射机制，它为实现用于存储资源管理的块设备驱动提供了一个高度模块化的内核架构，device mapper 的内核体系架构如图 24-6 所示。

device mapper 的工作原理是在内核中通过多个模块化的 target driver 插件实现对 IO 请求的过滤或重新定向等工作。当前已经实现的 target driver 插件包括软 raid、软加密、逻辑卷条带、多路径、镜像、快照、linear、mirror、snapshot、multipath 等。device mapper 进一步体现 Linux 内核设计中策略和机制分离的原则，将所有与策略相关的工作放到用户空间完成，内核中主要提供完成这些策略所需要的机制。

device mapper 用户空间相关部分主要负责配置具体的策略和控制逻辑，比如逻辑设备和哪些物理设备建立映射，如何建立映射关系等，而具体过滤和重新定向 IO 请求的工作由内核中相关代码完成。因此整个 device mapper 机制由两部分组成，分别是内核空间的

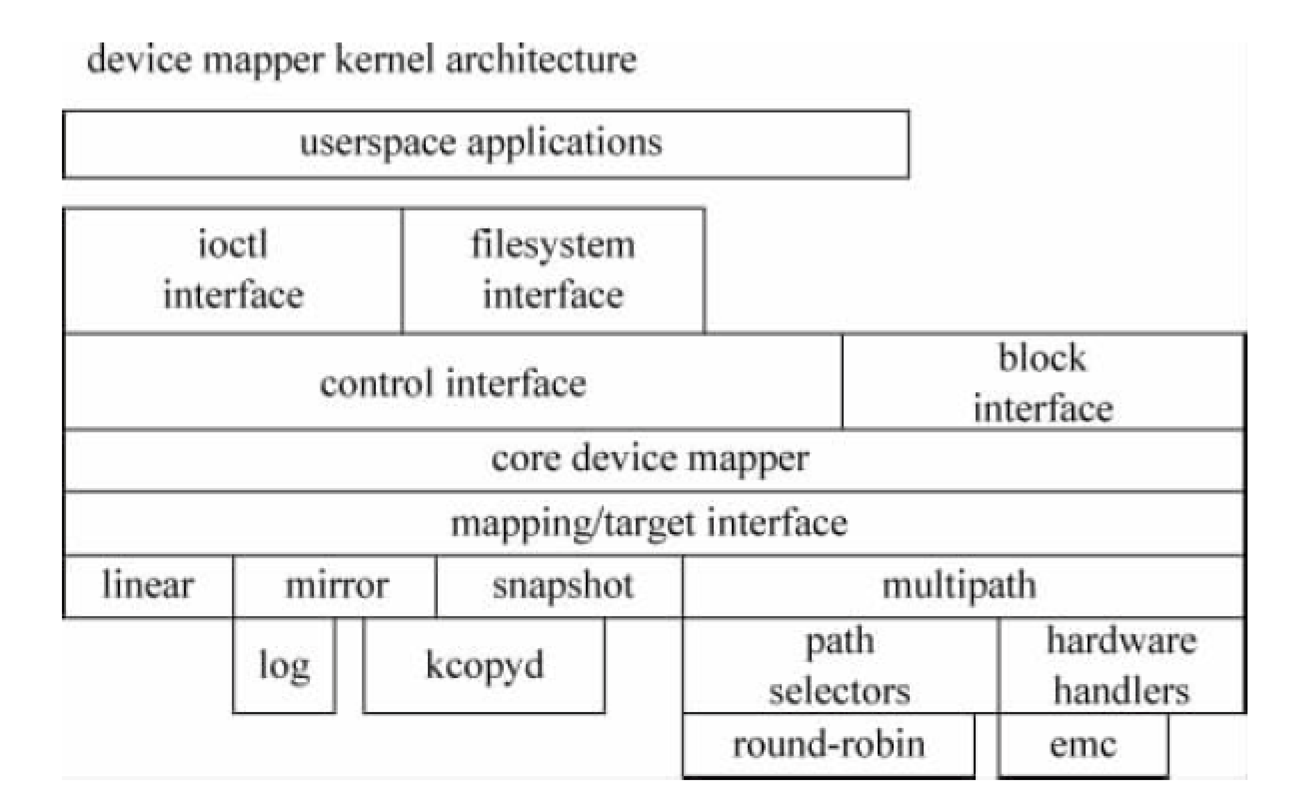


图 24-6　device mapper 文件系统

device mapper 驱动、用户空间的 device mapper 库以及它提供的 dmsetup 工具。

24.4　Docker 虚拟化特点

随着 Docker 在互联网企业中的不断应用，Docker 的优点越来越被 IT 人员认可，Docker 虚拟化跟传统虚拟化相比，有以下优点：

- 操作启动快，运行时的性能可以获取极大提升，管理操作（启动、停止、开始、重启等）都是以秒或毫秒为单位的；
- 轻量级虚拟化，用户会拥有足够的“操作系统”，仅需添加或减小镜像即可，单台服务器上可以部署 100～1000 个 containers 容器，而传统虚拟化能虚拟 10～20 个虚拟机就非常不错了；
- 开源免费，成本低，由现代 Linux 内核支持并驱动；
- 前景及云支持，正在越来越受欢迎，各大主流公司都在推动 Docker 的快速发展，性能有很大的优势；
- 更快速地交付和部署，Docker 在整个开发周期都可以完美的辅助用户实现快速交付；
- 更快速地创建及迭代，Docker 能够快速迭代应用程序，并让整个过程全程可见，使团队中的其他成员更容易理解应用程序是如何创建和工作的；
- 高效的部署和扩容，Docker 容器几乎可以在任意的平台上运行，包括物理机、虚拟机、公有云、私有云、个人电脑、服务器等；
- 更简单的管理，使用 Docker，只需要小小的修改，就可以替代以往大量的更新工作，所有的修改都以增量的方式被分发和更新，从而实现自动化并且高效的管理。

24.5 Docker 虚拟化原理

Docker 虚拟化中最核心部分为 Docker 引擎，Docker 引擎是一个 C/S(client/server)结构的应用，Docker 完整结构组件如图 24-7 所示。

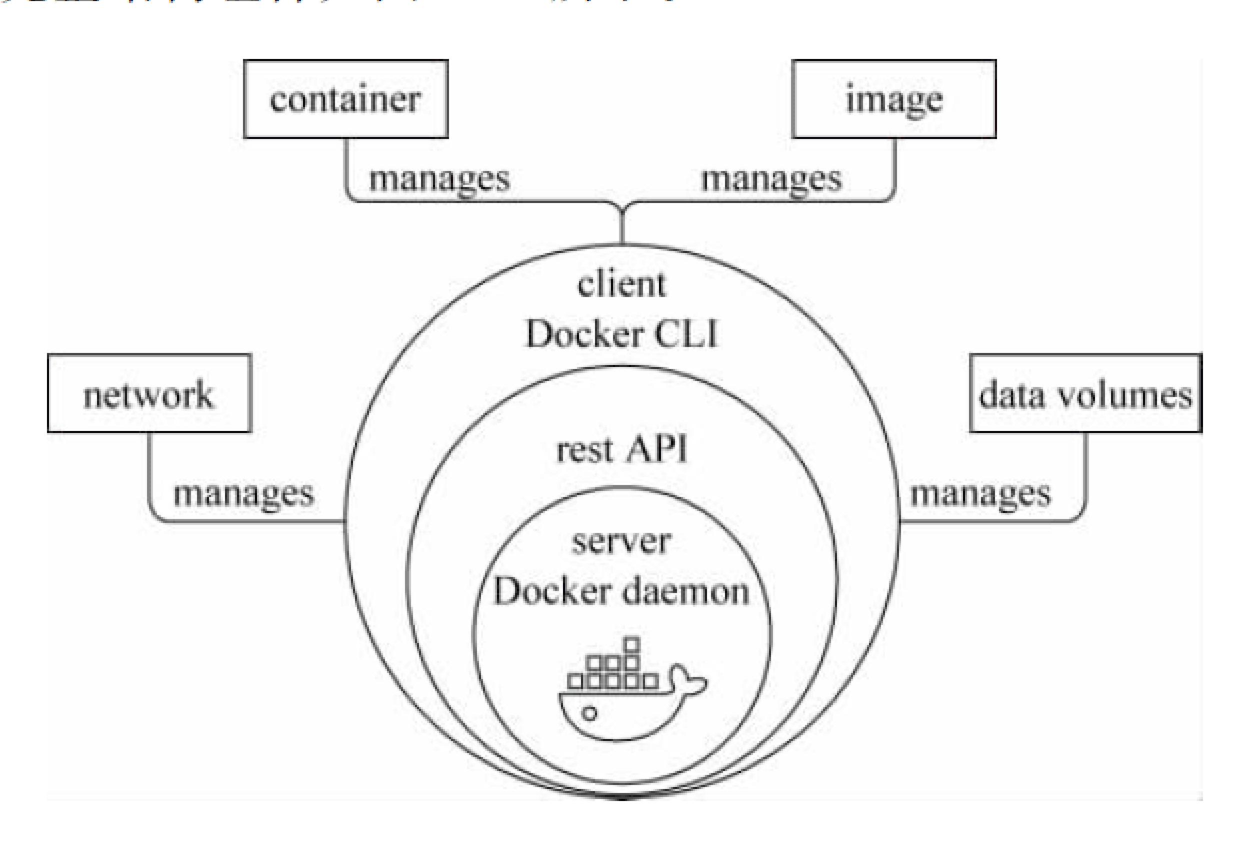


图 24-7　Docker 引擎结构

Docker server 是一个常驻进程，rest API 实现了 client 和 server 间的交互协议，CLI 实现容器和镜像的管理，为用户提供统一的操作界面。Docker 使用 C/S 架构，client 通过接口与 server 进程通信实现容器的构建、运行和发布，client 和 server 可以运行在同一台集群，也可以通过跨主机实现远程通信。

完整的 Docker 镜像可以支撑一个 Docker 容器的运行，在 Docker 容器运行过程中主要提供文件系统数据支撑。Docker 镜像作为 Docker 中最基本的概念，有以下特性：

- 镜像分层，每个镜像都由一个或多个镜像层组成；
- 可通过在某个镜像上加上一定的镜像层得到新镜像(此过程可通过编写 DockerFile 或基于容器 commit 实现)；
- 每个镜像层拥有唯一镜像 ID；
- 镜像在存储和使用时共享相同的镜像层(根据 ID)，所以在 pull 镜像时，已有的镜像层会自动跳过下载；
- 每个镜像层都是只读的，即使启动成容器，也无法对其真正的修改，修改只会作用于最上层的容器层。

Docker 容器，可以理解为一个或多个运行进程，而这些运行进程将占有相应的内存、相应的 CPU 计算资源、相应的虚拟网络设备以及相应的文件系统资源。而 Docker 容器所占用的文件系统资源，则通过 Docker 镜像的镜像层文件来提供。

基于每个镜像的 json 文件，Docker 可以通过解析 Docker 镜像的 json 文件，获知应该

在这个镜像之上运行什么样的进程,应该为进程配置怎样的环境变量,Docker 守护进程实现了静态向动态的转变。Docker 镜像层次如图 24-8 所示。

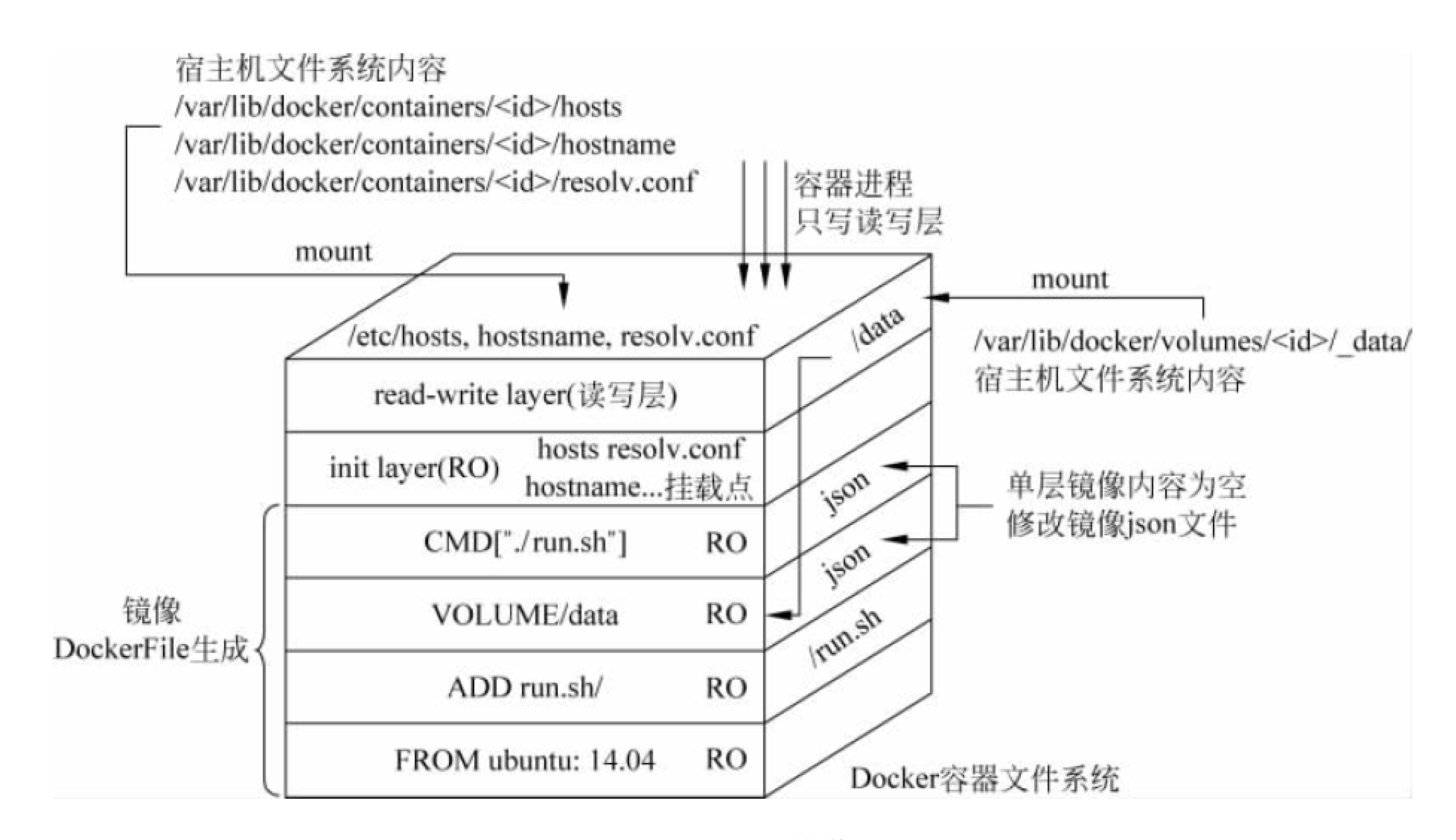


图 24-8 Docker 镜像分层

24.6 Docker 安装配置

Docker 官方文档要求部署 Docker 虚拟化的 Linux kernel 至少在 3.8 以上,即对应的 Linux 发行版为 CentOS 7.X,如果为 CentOS 6.X,那么 CentOS 6.X 应该如何安装呢?

CentOS 6.X 系列安装 Docker 软件,首先要关闭 SELinux,然后需要安装相应的 epel 源,安装代码如下,详细过程如图 24-9 所示。

```
sed -i '/SELINUX/s/enforcing/disabled/g' /etc/selinux/config
setenfoce 0
wget http://ftp.riken.jp/Linux/fedora/epel/6/x86_64/epel-release-6-8.noarch.rpm
rpm -ivh epel-release-6-8.noarch.rpm
yum install lxc libcgroup device-mapper-event-libs
yum install docker-io
yum install device-mapper* -y
```

Docker 安装完毕后,启动 docker 进程/etc/init.d/docker start ,并且查看 docker 进程 ps -ef |grep docker,如图 24-10 所示。

CentOS 7.X 系列安装 Docker 软件,首先要关闭 SELinux,然后需要安装相应的 epel

```
--2015-01-25 21:01:42--  http://ftp.riken.jp/Linux/fedo
.rpm
Resolving ftp.riken.jp... 134.160.38.1
Connecting to ftp.riken.jp|134.160.38.1|:80... connecte
HTTP request sent, awaiting response... 200 OK
Length: 14540 (14K) [text/plain]
Saving to: â€œepel-release-6-8.noarch.rpmâ€

100%[==============================================

2015-01-25 21:01:43 (66.8 KB/s) - â€œepel-release-6-8.r

[root@localhost ~]# rpm -ivh epel-release-6-8.noarch.rp
warning: epel-release-6-8.noarch.rpm: Header V3 RSA/SHA
Preparing...                ###########################
   1:epel-release           ###########################
```

(a) Docker安装epel源配置

```
Installed:
  lxc.x86_64 0:1.0.7-1.el6

Dependency Installed:
  lua-alt-getopt.noarch 0:0.7.0-1.el6             lua-file
  lua-lxc.x86_64 0:1.0.7-1.el6                    lxc-libs

Updated:
  libcgroup.x86_64 0:0.40.rc1-15.el6_6

Dependency Updated:
  libcgroup-devel.x86_64 0:0.40.rc1-15.el6_6

Complete!
[root@localhost ~]# yum install  lxc libcgroup
```

(b) Docker安装LXC配置

```
Installed:
  docker-io.x86_64 0:1.3.2-2.el6

Dependency Installed:
  bridge-utils.x86_64 0:1.2-10.el6

Dependency Updated:
  device-mapper.x86_64 0:1.02.90-2.el6_6.1    device-mapper-l

Complete!
[root@localhost ~]# yum install  docker-io
```

(c) Docker安装配置

图 24-9 Docker 安装配置

```
[root@localhost ~]#
[root@localhost ~]# /etc/init.d/docker start
Starting cgconfig service:                                 [  OK  ]
Starting docker:                                           [  OK  ]
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# ps -ef |grep docker
root      1476     1  3 21:04 pts/0    00:00:00 /usr/bin/docker -d
root      1585  1382  0 21:04 pts/0    00:00:00 grep docker
[root@localhost ~]#
```

图 24-10 Docker 服务启动

源，安装代码如下，详细过程如图 24-11 所示。

```
sed -i '/SELINUX/s/enforcing/disabled/g' /etc/selinux/config
setenfoce 0
yum install epel-release -y
yum install docker* -y
```

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# yum install docker* -y
Loaded plugins: fastestmirror
base
extras
updates
Loading mirror speeds from cached hostfile
 * base: mirror.bit.edu.cn
 * extras: mirrors.btte.net
 * updates: mirror.bit.edu.cn
Package docker-python is obsoleted by python-docker-py, trying to insta
0.6-1.el7.noarch instead
Package docker-registry is obsoleted by docker-distribution, trying to
tion-2.6.1-1.el7.x86_64 instead
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 2:1.12.6-32.git88a4867.el7.centos will be in
--> Processing Dependency: oci-systemd-hook >= 1:0.1.4-9 for package: 
```

(a) CentOS 7 Docker服务安装

```
[root@www-jfedu-net ~]# cat /etc/redhat-release
CentOS Linux release 7.2.1511 (Core)
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# systemctl start docker.service

[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ps -ef |grep docker
root      4314     1  2 02:35 ?        00:00:00 /usr/bin/dockerd-cur
runc=/usr/libexec/docker/docker-runc-current --default-runtime=docke
groupdriver=systemd --userland-proxy-path=/usr/libexec/docker/docke
nabled --log-driver=journald --signature-verification=false
root      4320  4314  0 02:35 ?        00:00:00 /usr/bin/docker-cont
var/run/docker/libcontainerd/docker-containerd.sock --shim docker-c
terval=0 --start-timeout 2m --state-dir /var/run/docker/libcontainer
ker-runc --runtime-args --systemd-cgroup=true
root      4483  4014  0 02:35 pts/0    00:00:00 grep --color=auto d
[root@www-jfedu-net ~]#
```

(b) CentOS 7启动Docker服务

```
[root@www-jfedu-net ~]# docker --help|more
Usage: docker [OPTIONS] COMMAND [arg...]
       docker [ --help | -v | --version ]

A self-sufficient runtime for containers.

Options:

  --config=~/.docker              Location of client config files
  -D, --debug                     Enable debug mode
  -H, --host=[]                   Daemon socket(s) to connect to
  -h, --help                      Print usage
  -l, --log-level=info            Set the logging level
  --tls                           Use TLS; implied by --tlsverify
  --tlscacert=~/.docker/ca.pem    Trust certs signed only by this C
  --tlscert=~/.docker/cert.pem    Path to TLS certificate file
  --tlskey=~/.docker/key.pem      Path to TLS key file
  --tlsverify                     Use TLS and verify the remote
```

(c) CentOS 7 Docker命令帮助

图 24-11　CentOS 7 Docker 安装

24.7　Docker 必备命令

对 Docker 技术的深入学习，需要构建 Docker 基础环境，熟练使用 Docker 各种语法命令，要模拟 Docker 虚拟化环境，需下载 Docker 镜像，通过命令在宿主机服务器上直接下载 Docker 公共仓库的镜像，具体步骤如下：

公共仓库 Nginx 和 CentOS 镜像下载以及本地导入 CentOS 镜像，执行如下命令，详细过程如图 24-12 所示。

```
docker pull nginx                               #Docker 下载 Nginx 镜像
docker pull centos                              #Docker 下载 CentOS 镜像
cat jfedu_centos68.tar |docker import - centos  #本地导入 CentOS 镜像
```

```
[root@www-jfedu-net ~]# docker pull nginx
latest: Pulling from nginx

5288d571feb1: Pulling fs layer
03b90b92b85e: Pulling fs layer
40f670bf3d4c: Download complete
4977170ffd18: Download complete
688223e95f40: Download complete
5226a97d5dde: Pulling fs layer
96fa29e2b3d5: Download complete
ce26041d2f2f: Download complete
171fcc793765: Pulling fs layer
71326c378a50: Download complete
```

(a) 官网下载Docker Nginx镜像

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# docker  pull centos
latest: Pulling from centos

d9cdac769794: Downloading [>
8b7794bcb4f9: Download complete
f3b88ddaed16: Download complete
```

(b) 官网下载Docker CentOS镜像

```
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# ll jfedu_centos68.tar
-rw-r--r-- 1 root root 668053504 Jan  5  2017 jfedu_centos68.tar
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# cat jfedu_centos68.tar |docker import - cent
sha256:ab80c6566251bd51fb00387f02fbb3eb2543ee1a6cd167b8bb2f4d7a6f452
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
centos              latest              ab80c6566251        8 second
[root@www-jfedu-net ~]#
[root@www-jfedu-net ~]#
```

(c) 本地导入Docker CentOS镜像

图 24-12　下载 Docker Nginx、CentOS 镜像及导入 CentOS 镜像

对 Docker 的管理除了可以下载镜像、导入镜像之外，还要掌握如下命令：

- docker version：查看 Docker 版本。
- docker search CentOS：搜索可用 Docker 镜像。
- docker images：查看当前 Docker 所有镜像。
- docker pull CentOS：下载 CentOS 镜像。
- cat xxx|docker import - newname：本地导入 Docker 镜像。
- docker export container_id > cenos6.tar：Docker 导出镜像。
- docker run CentOS echo "hello world"：在 Docker 容器中运行 hello world。
- docker run CentOS yum install ntpdate：在容器中安装 ntpdate 的程序。
- docker ps -l：获得最后一个容器的 ID。
- docker ps -a：查看所有的容器。
- docker commit 87e2313132 CentOS:v1：提交刚修改的容器。
- docker run -i -t -d CentOS /bin/bash：启动 Docker 镜像，-d 表示后台启动，-t 表示打开终端，-i 表示交互输入。
- Docker stop id：关闭 Docker 容器。

- Docker start id：启动 Docker 容器。
- docker rm id：删除 Docker 容器。
- docker rmi images：删除 Docker 镜像。
- docker run -d -p 80:80 -p 8022:22 CentOS:v2：-p 表示指定 Docker 容器端口映射，如 80:80，第一个 80 表示宿主机本地端口，第二个 80 表示 Docker 容器中端口，用户默认访问宿主机 80 端口，自动 NAT 映射到容器中 80 端口。
- docker exec -it docker_id /bin/bash：进入 Docker 容器 shell 终端。
- docker exec docker_id df -h：查看 Docker 容器内部磁盘分区。

24.8　Docker 网络详解

基于 Docker run 创建 Docker 容器时，可以使用--net 选项指定容器的网络模式，Docker 默认有以下 4 种网络模式：

- host 模式，使用--net=host 指定；
- container 模式，使用--net=container:NAME_or_ID 指定；
- none 模式，使用--net=none 指定；
- bridge 模式，使用--net=bridge 指定，默认设置。

Docker 4 种网络模式详解如下：

- host 模式详解，基于 host 模式，容器将不会获得一个独立的 network namespace，而是与宿主机共用一个 network namespace，容器将不会虚拟出自己的网卡、配置自己的 IP 等，而是使用宿主机的 IP 和端口。
- container 模式详解，理解 host 模式后，container 模式也非常好理解，container 模式指定新创建的容器和已经存在的一个容器共享一个 network namespace，而不是和宿主机共享。即新创建的容器不会创建自己的网卡、配置自己的 IP，而是和一个指定的容器共享 IP、端口范围等。同样两个容器除了网络方面相同之外，其他的如文件系统、进程列表等还是隔离的。
- none 模式详解，none 模式与其他的模式不同，如果 Docker 容器处于 none 模式，Docker 容器拥有自己的 network namespace，但是并不为 Docker 容器进行任何网络配置。该 Docker 容器没有网卡、IP、路由等信息，需要手工为 Docker 容器添加网卡、配置 IP 等，典型 pipework 工具为 Docker 容器指定 IP 等信息。
- bridge 桥接模式详解，bridge 模式是 Docker 默认的网络模式，该模式会为每一个容器分配 network namespace、设置 IP、路由等配置，默认会将 Docker 容器连接到一个虚拟网桥交换机 docker0 上，本文采用 bridge 模式，如图 24-13 所示为桥接模式拓扑图。

默认使用 Docker 命令创建 Docker 容器网络为 bridge 模式，以下为 Docker bridge 创建过程：

- 启动 Docker 容器，首先会在 Docker 宿主机上创建一对虚拟网卡 veth pair 设备，veth 设备总是成对出现的，组成了一条数据通道，数据从一端设备进入，就会从另一端设备出来，veth 设备常用来连接两个网络设备；
- Docker 将 veth pair 设备的一端放在新创建的容器中，并命名为 eth0，然后将另一端放在宿主机中，以 vethxxx 这样类似的名字命名，并将这个网络设备加入到 docker0 网桥中；
- 从 docker0 子网中分配一个 IP 给容器使用，并设置 docker0 的 IP 地址为容器的默认网关；

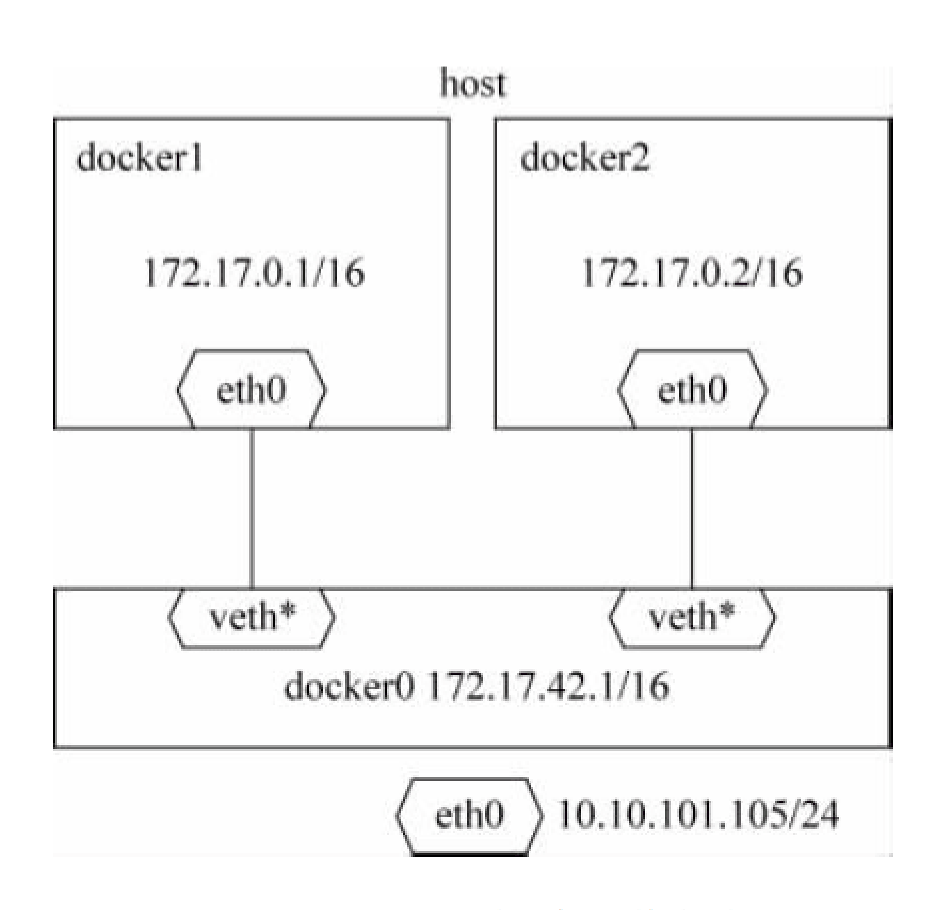


图 24-13　Docker 桥接网络拓扑图

- 此时，容器 IP 与宿主机能够通信，宿主机也可访问容器中的 IP 地址，在 bridge 模式下，连在同一网桥上的容器之间可以相互通信，同时容器也可以访问外网，但是其他宿主机不能访问 Docker 容器 IP，需要通过 NAT 将容器 IP 的 port 映射为宿主机的 IP 和 port，方可使用。

24.9　Docker 桥接配置

Docker 容器默认使用 docker0 桥接网络，IP 地址会自动分配，每个容器都是连接到 docker0 网桥上的。如果想让容器与宿主机同一网段的其他宿主机之间能访问，须在启动 Docker 的时候将 Docker 容器的某个端口映射到该宿主机的端口，其他宿主机连接 Docker 宿主机的 IP 和 port 即可。

在生产环境中可以自定义 Docker 桥接网卡，好处是可以设置 Docker 容器的 IP 与宿主机同网段，无须 NAT 映射端口访问，更加方便、快捷，同时也可以基于 pipework 脚本为 Docker 容器指定静态 IP 地址，以下为 Docker 自定义桥接网络的配置方法，执行代码如下：

```
yum install bridge-utils                  #安装 bridge 相关库支持
/etc/init.d/docker stop                   #停止 Docker 服务
ifconfig docker0 down                     #关掉 docker0
brctl delbr docker0                       #删除 docker
brctl addbr br0                           #创建 br0 网桥
ip link set dev br0 up                    #开启 br0 网桥
ip addr add 192.168.1.6/24 dev br0        #为 br0 分配物理网络中的 IP 地址
ip addr del 192.168.1.6/24 dev ens0       #将宿主机网卡的 IP 清空
```

```
brctl addif br0 eth0                  #将宿主机 eth0 网卡挂到 br0 上
ip route del default                  #删除原路由
ip route add default via 192.168.1.6 dev br0 #为 br0 设置路由
```

如果 Docker 宿主机操作系统为 CentOS 6.X，Docker 启用 br0 设置如下：

```
vim /etc/sysconfig/docker
other_args = " - b = br0"
```

上述配置方法比较烦琐，生产环境建议直接通过创建网桥 br0 配置文件实现桥接，在 /etc/sysconfig/network-scripts/下，修改原 ifcfg-eth0 网卡配置，同时增加 ifcfg-br0 桥接网卡配置，操作步骤如下。

（1）vi ifcfg-eth0 内容修改为如下：

```
DEVICE = eth0
BOOTPROTO = none
NM_CONTROLLED = no
ONBOOT = yes
TYPE = Ethernet
BRIDGE = "br0"
IPADDR = 192.168.1.6
NETMASK = 255.255.255.0
GATEWAY = 192.168.1.254
USERCTL = no
```

（2）vi ifcfg-br0 内容如下：

```
DEVICE = "br0"
BOOTPROTO = none
IPV6INIT = no
NM_CONTROLLED = no
ONBOOT = yes
TYPE = "Bridge"
IPADDR = 192.168.1.6
NETMASK = 255.255.255.0
GATEWAY = 192.168.1.254
USERCTL = no
```

（3）Docker 桥接网卡配置完毕，直接重启 network 服务即可。

```
/etc/init.d/network restart
```

（4）修改 Docker 桥接网卡为 br0。如果 Docker 宿主机操作系统为 CentOS 7.X，Docker 启用 br0 设置如下，然后重启 Docker 服务即可，最终网卡配置信息如图 24-14 所示。

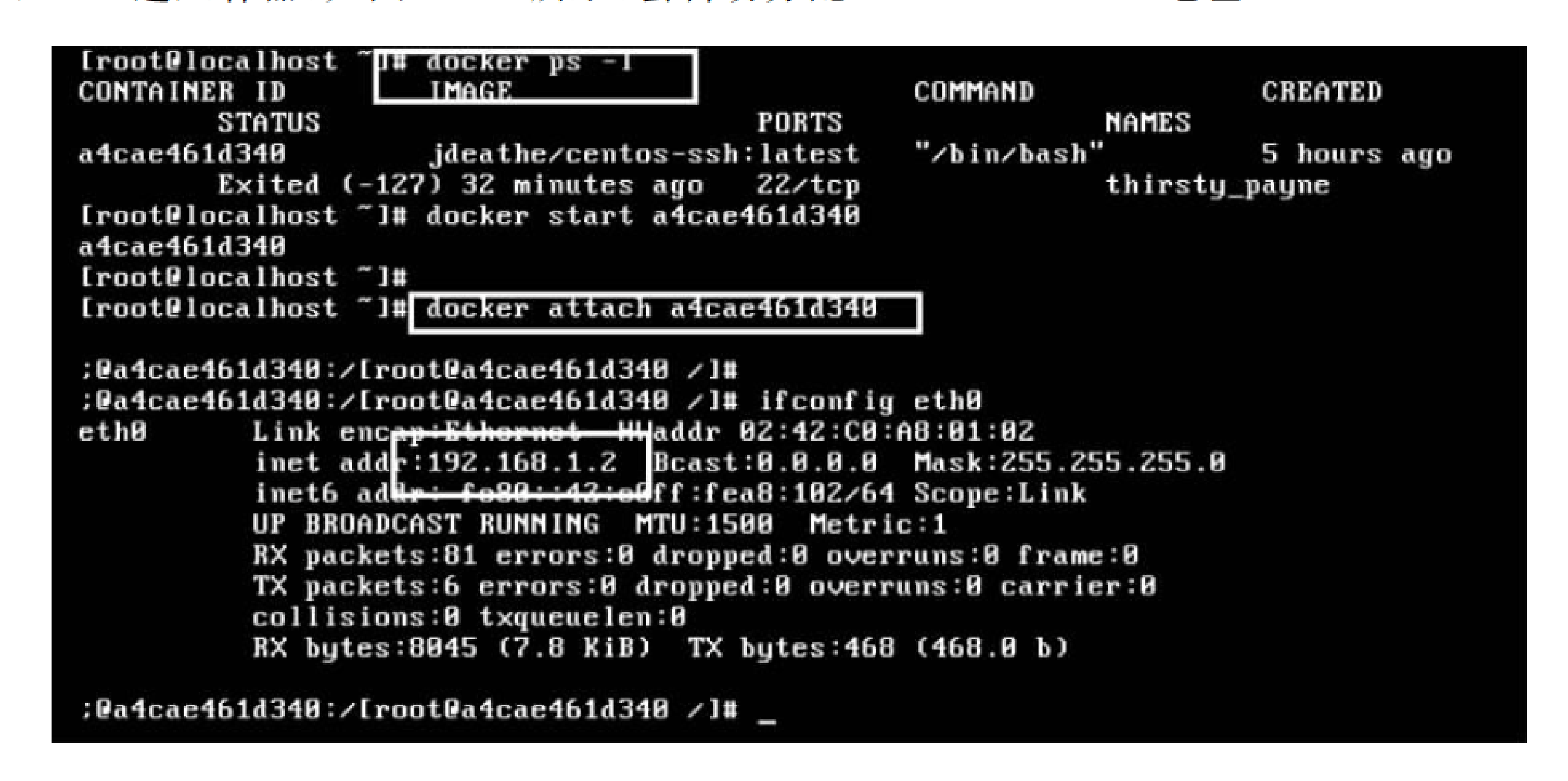
```
vim /etc/sysconfig/docker - network
DOCKER_NETWORK_OPTIONS = " - b = br0"
```

```
br0       Link encap:Ethernet  HWaddr 00:0C:29:67:FA:D3
          inet addr:192.168.1.6  Bcast:192.168.1.255  Mask:255.255.
          inet6 addr: fe80::20c:29ff:fe67:fad3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:165 errors:0 dropped:0 overruns:0 frame:0
          TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:41513 (40.5 KiB)  TX bytes:6286 (6.1 KiB)

eth0      Link encap:Ethernet  HWaddr 00:0C:29:67:FA:D3
          inet6 addr: fe80::20c:29ff:fe67:fad3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7943 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4158 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1168582 (1.1 MiB)  TX bytes:663488 (647.9 KiB)
          Interrupt:19 Base address:0x2000
```

图 24-14　宿主机网桥 br0 配置

启动新 Docker 容器，使用命令 docker attach 容器 ID 或 docker　exec　-it 容器 ID /bin/bash 进入容器，如图 24-15 所示，会自动分配 192.168.1.2 IP 地址。



```
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE                       COMMAND             CREATED
             STATUS                      PORTS               NAMES
a4cae461d340        jdeathe/centos-ssh:latest   "/bin/bash"         5 hours ago
             Exited (-127) 32 minutes ago   22/tcp              thirsty_payne
[root@localhost ~]# docker start a4cae461d340
a4cae461d340
[root@localhost ~]#
[root@localhost ~]# docker attach a4cae461d340

;@a4cae461d340:/[root@a4cae461d340 /]#
;@a4cae461d340:/[root@a4cae461d340 /]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 02:42:C0:A8:01:02
          inet addr:192.168.1.2  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::42:c0ff:fea8:102/64 Scope:Link
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:81 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:8045 (7.8 KiB)  TX bytes:468 (468.0 b)

;@a4cae461d340:/[root@a4cae461d340 /]# _
```

图 24-15　Docker 容器获取 br0 网段 IP

通过配置 br0 桥接网卡，快速实现 Docker 容器快速获取动态 IP 地址，生产环境服务器的 IP 均为静态 IP，基于 pipework 工具为 Docker 容器指定静态 IP 地址，以下为 pipework 工具配置 Docker 容器静态 IP 地址的方法，通过 pipework 指定的静态 IP，当容器重启之后，静态 IP 会丢失，所以启动容器之前需重新指定该 IP，也可以通过 shell 脚本自动配置 IP，代码如下：

```
#安装 pipework 工具
git clone https://github.com/jpetazzo/pipework
cp ~/pipework/pipework /usr/local/bin/
#启动新的 Docker 容器，网络设置为 none，名称设置为 lamp2
```

```
docker run -itd --net=none --name=lamp2 CentOS 7 /bin/bash
#基于 pipework 设置 Docker 容器 IP 为 192.168.1.11,网关为 192.168.1.6,Docker 容器 IP 子网掩
#码为 255.255.255.0
pipework br0 lamp2 192.168.1.11/24@192.168.1.6
```

查看 Docker 容器 IP 地址，执行代码如下，结果如图 24-16 所示。

```
docker exec lamp2 ifconfig
```

```
bash-4.1# ifconfig
eth1      Link encap:Ethernet  HWaddr FA:B9:11:74:4D:07
          inet addr:192.168.1.11  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::f8b9:11ff:fe74:4d07/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1007 errors:0 dropped:0 overruns:0 frame:0
          TX packets:43 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:192809 (188.2 KiB)  TX bytes:3574 (3.4 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

图 24-16　Docker 容器 pipework 静态配置 IP

24.10　DockerFile 参数详解

DockFile 是一种能被 Docker 程序解释的脚本，DockerFile 由多条指令组成，每条指令对应 Linux 系统中不同的命令，基于 DockerFile 可以自定义创建生产环境所需的 Docker 镜像，通过镜像可以启动所需的 Docker 容器。

Docker 程序将这些 DockerFile 指令翻译为真正的 Linux 命令，DockerFile 有特定的书写格式和支持的命令，Docker 程序解决这些命令间的依赖关系，类似于 Linux 系统中编译软件所使用的 MakeFile 文件。

Docker 程序可以读取 DockerFile 文件，根据指令生成定制的 image，需要定制自己额外的需求时，只需在 DockerFile 上添加或修改指令，重新生成 image 即可，省去了敲命令的麻烦，以下为 DockerFile 镜像制作常用的命令详解：

- FROM <image>:<tag>：FROM 指令表示指定一个基本的镜像源，或从公共库拉取一个镜像源，DockerFile 文件第一行必须指定 FROM 基础镜像源。
- MAINTAINER：设置 DockerFile 编写人或维护者的信息。
- LABEL <key>=<value>：设置标签，采用键值对的形式。
- RUN <command>：核心指令，表示运行的 Linux 指令，每条 RUN 指令在当前基础镜像上执行，并且提交成为新的镜像。
- EXPOSE <port>[<port>]：用来指定 Docker 容器中监听的端口，用于外界宿主机互联访问，启动 Docker 时，可以通过-P，主机会自动分配一个端口号转发到指定的

端口。

- ENV < key >=< value >：设置环境变量，执行 RUN 指令及 Docker 启动时被引用。
- WORKDIR /path/to/workdir：设置工作目录，执行 RUN、ADD、COPY、ENV 指令时的基础路径。
- COPY < src > < dest >和 ADD < src > < dest >：Linux 系统增加及复制文件，ADD 在和 COPY 相同的基础上，ADD 允许< src >是一个 URL，同时 ADD 的< src >是一个压缩格式文档，< src >将会解压缩复制。
- CMD 和 ENTRYPOINT：配置 Docker 容器启动后执行的命令，每个 DockerFile 至少指定一个 CMD 命令或 ENTRYPOINT，两者都可以指定 shell 或 exec 函数调用的方式执行命令，默认 DockerFile run 启动镜像之后便会退出容器，需要一个长时间运行的命令，使得容器一直执行。

CMD 和 ENTRYPOINT 的详解如下。

- CMD ["executable","param1","param2"]：运行一个可执行的文件并提供参数。
- CMD ["param1","param2"]：为 ENTRYPOINT 指定参数。
- CMD command param1 param2 ：以/bin/sh -c 的方法执行的命令。
- ENTRYPOINT ["executable", "param1", "param2"] ：首选执行形式。
- ENTRYPOINT command param1 param2：以/bin/sh -c 的方法执行的命令。

CMD 和 ENTRYPOINT 的区别如下：

每个 DockerFile 只能有一个 CMD/ENTRYPOINT 指令，超过一个 CMD 只有最后一个生效；

CMD 在运行时会被 Docker run command 指定命令覆盖，而 ENTRYPOINT 不会被运行时 Docker run command 覆盖；

DockerFile 中同时设置 CMD 和 ENTRYPOINT，Docker 在 build 过程中会将 CMD 中指定的内容作为 ENTRYPOINT 的参数；

如果 Docker 启动需运行多个启动命令，彼此之间可以使用 && 分开，最后一个命令必须为无限运行的命令，否则启动的容器将会被退出。

- VOLUME [DIR] ：设置本地挂载目录，用于存放数据库和需要保持的数据。
- USER daemon：指定 Docker 运行时的用户名或 UID，后续的 RUN 也会使用指定用户。
- ONBUILD [INSTRUCTION]：配置当前所创建的镜像作为其他新创建镜像的基础镜像时，所执行的操作指令。

24.11 DockerFile 企业案例一

基于 DockerFile 相关指令，可以在 Docker 宿主机上编写 DockerFile 文件，本案例为实现 Docker 容器运行，并对外开启 22 端口，DockerFile 代码如下：

```
# 设置基本的镜像,后续命令都以这个镜像为基础
FROM CentOS_lamp:v1
# 作者信息
MAINTAINER JFEDU.NET
# RUN 命令会在上面指定的镜像里执行任何命令
RUN yum install passwd openssl openssh-server -y
RUN echo '123456' | passwd --stdin root
RUN sed -i '/^session\s\+required\s\+pam_loginuid.so/s/^/#/' /etc/pam.d/sshd
RUN mkdir -p /root/.ssh && chown root.root /root && chmod 700 /root/.ssh
RUN mkdir /var/run/sshd
#暴露 ssh 端口 22
EXPOSE 22
# 设定运行镜像时的默认命令并且打印 Docker IP 地址,以 daemon 方式启动 sshd
CMD ip addr ls eth0 | awk '{print $2}' | egrep -o '([0-9]+\.){3}[0-9]+';/usr/sbin/sshd -D
```

24.12　DockerFile 企业案例二

基于 DockerFile 相关指令，可以在 Docker 宿主机上编写 DockerFile 文件，本案例为实现 Docker 容器运行，并对外开启 80 端口，DockerFile 代码如下：

```
# 设置基本的镜像,后续命令都以这个镜像为基础
FROM CentOS_lamp:v1
# 作者信息
MAINTAINER JFEDU.NET
# RUN 命令会在上面指定的镜像里执行任何命令
RUN yum install pcre-devel -y
RUN yum install httpd httpd-devel -y
RUN echo "<h1>The Test Page JFEDU</h1>" >>/var/www/html/index.html
#暴露 ssh 端口 80
EXPOSE 80
#启动 httpd
CMD ["/usr/sbin/apachectl", "-D", "FOREGROUND"]
```

24.13　DockerFile 企业案例三

基于 DockerFile 相关指令，可以在 Docker 宿主机上编写 DockerFile 文件，本案例为实现 Docker 容器运行，并对外开启 3306 端口，DockerFile 代码如下：

```
FROM CentOS:v1
RUN groupadd -r mysql && useradd -r -g mysql mysql
RUN install -y gcc zlib-devel gd-devel
ENV MYSQL_MAJOR 5.6
ENV MYSQL_VERSION 5.6.20
RUN
```

```
    && curl -SL "http://dev.mysql.com/get/Downloads/MySQL-$MYSQL_MAJOR/mysql-$MYSQL_VERSION-linux-glibc2.5-x86_64.tar.gz" -o mysql.tar.gz \
    && curl -SL "http://mysql.he.net/Downloads/MySQL-$MYSQL_MAJOR/mysql-$MYSQL_VERSION-linux-glibc2.5-x86_64.tar.gz.asc" -o mysql.tar.gz.asc \
    && mkdir /usr/local/mysql \
    && tar -xzf mysql.tar.gz -C /usr/local/mysql \
    && rm mysql.tar.gz* \
ENV PATH $PATH:/usr/local/mysql/bin:/usr/local/mysql/scripts
WORKDIR /usr/local/mysql
VOLUME /var/lib/mysql
EXPOSE 3306
CMD ["mysqld", "--datadir=/var/lib/mysql", "--user=mysql"]
```

24.14 DockerFile 企业案例四

基于 DockerFile 相关指令，可以在 Docker 宿主机上编写 DockerFile 文件，本案例为实现 Docker 容器运行，并对外开启 8080 端口，DockerFile 代码如下：

```
FROM CentOS:v1
#设置 DockerFile 运行工作目录
WORKDIR /tmp
#安装 JAVA JDK
RUN wget --no-cookies --no-check-certificate --header "Cookie:gpw_e24=http%3a%2f%2fwww.oracle.com%2ftechnetwork%2fjava%2fjavase%2fdownloads%2fjdk7-downloads-1880260.html;oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/7u79-b15/jdk-7u79-linux-x64.tar.gz
RUN tar -zxf jdk-7u79-linux-x64.tar.gz
RUN mkdir -p /usr/java/
RUN mv jdk1.7.0_79 /usr/java/
#配置环境变量
ENV JAVA_HOME /usr/java/jdk1.7.0_79/
ENV JRE_HOME $JAVA_HOME/jre
ENV CLASSPATH .:$JAVA_HOME/lib:$JRE_HOME/lib
ENV PATH $PATH:$JAVA_HOME/bin
#安装配置 tomcat 服务
RUN wget http://mirror.bit.edu.cn/apache/tomcat/tomcat-7/v7.0.62/bin/apache-tomcat-7.0.62.tar.gz
RUN tar xvf apache-tomcat-7.0.62.tar.gz
RUN mv apache-tomcat-7.0.62 /usr/local/tomcat/
#配置 tomcat 环境变量
ENV CATALINA_HOME /usr/local/tomcat/
EXPOSE 8080
#设置 tomcat 自启动
CMD [ "/usr/local/tomcat/bin/catalina.sh", "run" ]
```

24.15 Docker磁盘扩容

device mapper是Linux 2.6内核中提供的一种从逻辑设备到物理设备的映射框架机制，device mapper driver默认会创建一个100GB的存储文件，主要用于存储镜像和容器，每一个容器都被限制在10GB大小的卷内，也可以基于loopback自动创建稀疏文件，具体为用/var/lib/docker/devicemapper/devicemapper下的data和metadata实现动态磁盘扩容，默认创建的100GB存储总空间和Docker容器10GB空间是无法满足生产环境应用的，需要扩大Docker总容量和Docker容器的rootfs根系统大小。

Docker服务在启动的时候可以配置device mapper的启动参数，docker -d --storage-opt dm.foo=bar，常见参数如下。

- dm.basesize：默认为10GB，限制容器和镜像的大小。
- dm.loopdatasize：存储池大小，默认为100GB。
- dm.datadev：存储池设备，/var/lib/docker/devicemapper/devicemapper/data。
- dm.loopmetadatasize：元数据大小，默认为2GB。
- dm.metadatadev：元数据设备，/var/lib/docker/devicemapper/devicemapper/metadata。
- dm.fs：文件系统，默认为ext4。
- dm.blocksize blocksize：默认为64KB。
- dm.blkdiscard：默认为true。

将Docker默认存储池从100GB扩大到2TB，存储池元数据从2GB扩大到10GB，执行命令如下：

```
rm -rf /var/lib/docker/devicemapper/devicemapper
mkdir -p /var/lib/docker/devicemapper/devicemapper
dd if=/dev/zero of=/var/lib/docker/devicemapper/devicemapper/data bs=1G count=0 seek=2000
dd if=/dev/zero of=/var/lib/docker/devicemapper/devicemapper/metadata bs=1G count=0 seek=10
```

还可以通过配置文件直接添加以下代码实现将Docker默认存储池从100GB扩大到2TB，存储池元数据从2GB扩大到10GB，修改配置文件/etc/sysconfig/docker-storage，加入如下代码：

```
DOCKER_STORAGE_OPTIONS="--storage-opt dm.loopdatasize=2000G --storage-opt dm.
loopmetadatasize=10G --storage-opt dm.fs=ext4"
```

上述配置完毕后，重启Docker服务即可，新生成的Docker容器磁盘大小即可生效，如果不在配置文件中指定，还可以基于以下命令直接启动，生产环境推荐修改配置文件，而不推荐命令行方式直接启动，执行命令如下：

```
docker -d --storage-opt dm.loopdatasize=2000G --storage-opt dm.loopmetadatasize=10G
--storage-opt dm.fs=ext4
```

以上方法只适用于新容器生成，并且修改后需要重启 Docker，无法做到动态地给正在运行的容器指定大小，基于现有容器在线扩容，宿主机文件系统类型支持 ext2、ext3、ext4，不支持 XFS。以下为 Docker 在线扩容的方法。

(1) 查看原容器的磁盘空间大小，如图 24-17 所示。

```
Discuz_X3.2_SC_UTF8.zip  index.html
[root@449c143b14b1 ~]# df -h
文件系统                容量  已用  可用 已用% 挂载点
rootfs                  9.8G  588M  8.7G    7% /
/dev/mapper/docker-8:2-1704214-449c143b14b12e0880716 5602eff78c
                        9.8G  588M  8.7G    7% /
tmpfs                   497M     0  497M    0% /dev
shm                      64M     0   64M    0% /dev/shm
tmpfs                   497M     0  497M    0% /run
tmpfs                   497M     0  497M    0% /tmp
/dev/sda2                29G  3.4G   24G   13% /etc/resolv.conf
/dev/sda2                29G  3.4G   24G   13% /etc/hostname
/dev/sda2                29G  3.4G   24G   13% /etc/hosts
tmpfs                   497M     0  497M    0% /run/secrets
tmpfs                   497M     0  497M    0% /proc/kcore
[root@449c143b14b1 ~]# exit
```

图 24-17　Docker 容器默认 rootfs 扩容

(2) 查看 Docker 存储 device mapper 设备名称，如图 24-18 所示。

```
[root@localhost ~]#
[root@localhost ~]#  ls -l /dev/mapper/docker-*
lrwxrwxrwx. 1 root root 7 5月  28 12:47 /dev/mapper/docker-8:2-1704214-449c143b14b12e088
2eff78ca2e39a5f444fb638b3d7131ccaa3d4843 -> ../dm-1
lrwxrwxrwx. 1 root root 7 5月  28 12:45 /dev/mapper/docker-8:2-1704214-pool -> ../dm-0
[root@localhost ~]#
```

图 24-18　Docker 容器对应的 device mapper 设备

(3) 查看 Docker mapper 卷信息表，如图 24-19 所示。

```
[root@localhost ~]#  ls -l /dev/mapper/docker-*
lrwxrwxrwx. 1 root root 7 5月  28 12:47 /dev/mapper/docker-8:2-1704214-449c143b14b12e0880716
2eff78ca2e39a5f444fb638b3d7131ccaa3d4843 -> ../dm-1
lrwxrwxrwx. 1 root root 7 5月  28 12:45 /dev/mapper/docker-8:2-1704214-pool -> ../dm-0
[root@localhost ~]# dmsetup table docker-8:2-1704214-449c143b14b12e0880716 5602eff78ca2e39a5f4
fb638b3d7131ccaa3d4843
0 20971520 thin 253:0 30
[root@localhost ~]#
```

图 24-19　Docker 容器对应的 mapper 信息表

(4) 计算扩容扇区大小。

如图 24-19 所示 20971520 数字表示设备的大小，表示有多少个 512B 的扇区。这个值略高于 10GB 的大小，将原 10GB 的空间扩容为 15GB，计算 15GB 的空间所需扇区的大小，计算命令如下：

```
echo $((15 * 1024 * 1024 * 1024/512))
31457280
```

然后修改 Docker 容器卷信息表、激活并且验证，使用 echo 命令将新的扇区大小写入，

注意只是改变 20971520 的数字为 31457280，其他数字不变，通过命令 dmsetup resume 将修改后的容器文件激活，通过命令 dmsetup table 查看最新 Docker 扇区信息，如图 24-20 所示。

```
0 20971520 thin 253:0 30
[root@localhost ~]# echo 0 31457280 thin 253:0 30| dmsetup load docker-8:2-1704214-449c143b14b1
2e08807165602eff78ca2e39a5f444fb638b3d7131ccaa3d4843
[root@localhost ~]#
[root@localhost ~]# dmsetup resume docker-8:2-1704214-449c143b14b12e08807165602eff78ca2e39a5f44
4fb638b3d7131ccaa3d4843
[root@localhost ~]#
[root@localhost ~]# dmsetup table  docker-8:2-1704214-449c143b14b12e08807165602eff78ca2e39a5f44
4fb638b3d7131ccaa3d4843
0 31457280 thin 253:0 30
[root@localhost ~]#
```

图 24-20　Docker 容器扩容过程

（5）修改文件系统大小，命令为 resize2fs，如图 24-21 所示。

```
[root@localhost ~]# resize2fs /dev/mapper/docker-8:2-1704214-449c143b14b12e
39a5f444fb638b3d7131ccaa3d4843
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/mapper/docker-8:2-1704214-449c143b14b12e08807165602eff78
7131ccaa3d4843 is mounted on /var/lib/docker/devicemapper/mnt/449c143b14b12
e39a5f444fb638b3d7131ccaa3d4843; on-line resizing required
old_desc_blocks = 2, new_desc_blocks = 2
The filesystem on /dev/mapper/docker-8:2-1704214-449c143b14b12e08807165602e
8b3d7131ccaa3d4843 is now 3932160 blocks long.

[root@localhost ~]#
```

图 24-21　Docker 容器设备扩容

（6）验证 Docker rootfs 磁盘大小，如图 24-22 所示。

```
[root@localhost ~]# docker attach 449c143b14b1

[root@449c143b14b1 /]#
[root@449c143b14b1 /]#
[root@449c143b14b1 /]#
[root@449c143b14b1 /]# df -h
Filesystem             Size  Used Avail Use% Mounted on
rootfs                  15G  592M   14G   5% /
/dev/mapper/docker-8:2-1704214-449c143b14b12e08807165602eff78ca2e39a5
                        15G  592M   14G   5% /
tmpfs                  497M     0  497M   0% /dev
shm                     64M     0   64M   0% /dev/shm
tmpfs                  497M     0  497M   0% /run
tmpfs                  497M     0  497M   0% /tmp
/dev/sda2               29G  3.4G   24G  13% /etc/resolv.conf
/dev/sda2               29G  3.4G   24G  13% /etc/hostname
/dev/sda2               29G  3.4G   24G  13% /etc/hosts
tmpfs                  497M     0  497M   0% /run/secrets
tmpfs                  497M     0  497M   0% /proc/kcore
```

图 24-22　Docker 容器扩容为 15GB

通过上述步骤成功地将 Docker 容器的 10GB 空间扩容为 15GB，还可以将上述步骤写成 shell 脚本，基于脚本参数快速扩容。给 Docker 磁盘扩容除了采用以上方法外，还可以使用挂载目录方法，基于-v 参数，在启动 Docker 容器时指定。

24.16　Docker 构建私有仓库

Docker 镜像默认存放在仓库中，Docker 仓库分为公共仓库和私有仓库，随着公司业务的发展，Docker 镜像的种类也非常繁多，为了统一管理，可以基于 registry 搭建本地私有

注意只是改变 20971520 的数字为 31457280，其他数字不变，通过命令 dmsetup resume 将修改后的容器文件激活，通过命令 dmsetup table 查看最新 Docker 扇区信息，如图 24-20 所示。

```
0 20971520 thin 253:0 30
[root@localhost ~]# echo 0 31457280 thin 253:0 30| dmsetup load docker-8:2-1704214-449c143b14b1
2e08807165602eff78ca2e39a5f444fb638b3d7131ccaa3d4843
[root@localhost ~]#
[root@localhost ~]# dmsetup resume docker-8:2-1704214-449c143b14b12e08807165602eff78ca2e39a5f44
4fb638b3d7131ccaa3d4843
[root@localhost ~]#
[root@localhost ~]# dmsetup table  docker-8:2-1704214-449c143b14b12e08807165602eff78ca2e39a5f44
4fb638b3d7131ccaa3d4843
0 31457280 thin 253:0 30
[root@localhost ~]#
```

图 24-20　Docker 容器扩容过程

（5）修改文件系统大小，命令为 resize2fs，如图 24-21 所示。

```
[root@localhost ~]# resize2fs /dev/mapper/docker-8:2-1704214-449c143b14b12e
39a5f444fb638b3d7131ccaa3d4843
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/mapper/docker-8:2-1704214-449c143b14b12e08807165602eff78
7131ccaa3d4843 is mounted on /var/lib/docker/devicemapper/mnt/449c143b14b12
e39a5f444fb638b3d7131ccaa3d4843; on-line resizing required
old_desc_blocks = 2, new_desc_blocks = 2
The filesystem on /dev/mapper/docker-8:2-1704214-449c143b14b12e08807165602e
8b3d7131ccaa3d4843 is now 3932160 blocks long.

[root@localhost ~]#
```

图 24-21　Docker 容器设备扩容

（6）验证 Docker rootfs 磁盘大小，如图 24-22 所示。

```
[root@localhost ~]# docker attach 449c143b14b1

[root@449c143b14b1 /]#
[root@449c143b14b1 /]#
[root@449c143b14b1 /]#
[root@449c143b14b1 /]# df -h
Filesystem          Size  Used Avail Use% Mounted on
rootfs               15G  592M   14G   5% /
/dev/mapper/docker-8:2-1704214-449c143b14b12e08807165602eff78ca2e39a5
                     15G  592M   14G   5% /
tmpfs               497M     0  497M   0% /dev
shm                  64M     0   64M   0% /dev/shm
tmpfs               497M     0  497M   0% /run
tmpfs               497M     0  497M   0% /tmp
/dev/sda2            29G  3.4G   24G  13% /etc/resolv.conf
/dev/sda2            29G  3.4G   24G  13% /etc/hostname
/dev/sda2            29G  3.4G   24G  13% /etc/hosts
tmpfs               497M     0  497M   0% /run/secrets
tmpfs               497M     0  497M   0% /proc/kcore
```

图 24-22　Docker 容器扩容为 15GB

通过上述步骤成功地将 Docker 容器的 10GB 空间扩容为 15GB，还可以将上述步骤写成 shell 脚本，基于脚本参数快速扩容。给 Docker 磁盘扩容除了采用以上方法外，还可以使用挂载目录方法，基于-v 参数，在启动 Docker 容器时指定。

24.16　Docker 构建私有仓库

Docker 镜像默认存放在仓库中，Docker 仓库分为公共仓库和私有仓库，随着公司业务的发展，Docker 镜像的种类也非常繁多，为了统一管理，可以基于 registry 搭建本地私有

注意只是改变 20971520 的数字为 31457280，其他数字不变，通过命令 dmsetup resume 将修改后的容器文件激活，通过命令 dmsetup table 查看最新 Docker 扇区信息，如图 24-20 所示。

```
0 20971520 thin 253:0 30
[root@localhost ~]# echo 0 31457280 thin 253:0 30| dmsetup load docker-8:2-1704214-449c143b14b1
2e08807165602eff78ca2e39a5f444fb638b3d7131ccaa3d4843
[root@localhost ~]#
[root@localhost ~]# dmsetup resume docker-8:2-1704214-449c143b14b12e08807165602eff78ca2e39a5f44
4fb638b3d7131ccaa3d4843
[root@localhost ~]#
[root@localhost ~]# dmsetup table  docker-8:2-1704214-449c143b14b12e08807165602eff78ca2e39a5f44
4fb638b3d7131ccaa3d4843
0 31457280 thin 253:0 30
[root@localhost ~]#
```

图 24-20　Docker 容器扩容过程

（5）修改文件系统大小，命令为 resize2fs，如图 24-21 所示。

```
[root@localhost ~]# resize2fs /dev/mapper/docker-8:2-1704214-449c143b14b12e
39a5f444fb638b3d7131ccaa3d4843
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/mapper/docker-8:2-1704214-449c143b14b12e08807165602eff78
7131ccaa3d4843 is mounted on /var/lib/docker/devicemapper/mnt/449c143b14b12
e39a5f444fb638b3d7131ccaa3d4843; on-line resizing required
old_desc_blocks = 2, new_desc_blocks = 2
The filesystem on /dev/mapper/docker-8:2-1704214-449c143b14b12e08807165602e
8b3d7131ccaa3d4843 is now 3932160 blocks long.

[root@localhost ~]#
```

图 24-21　Docker 容器设备扩容

（6）验证 Docker rootfs 磁盘大小，如图 24-22 所示。

```
[root@localhost ~]# docker attach 449c143b14b1

[root@449c143b14b1 /]#
[root@449c143b14b1 /]#
[root@449c143b14b1 /]#
[root@449c143b14b1 /]# df -h
Filesystem          Size  Used Avail Use% Mounted on
rootfs               15G  592M   14G   5% /
/dev/mapper/docker-8:2-1704214-449c143b14b12e08807165602eff78ca2e39a5
                     15G  592M   14G   5% /
tmpfs               497M     0  497M   0% /dev
shm                  64M     0   64M   0% /dev/shm
tmpfs               497M     0  497M   0% /run
tmpfs               497M     0  497M   0% /tmp
/dev/sda2            29G  3.4G   24G  13% /etc/resolv.conf
/dev/sda2            29G  3.4G   24G  13% /etc/hostname
/dev/sda2            29G  3.4G   24G  13% /etc/hosts
tmpfs               497M     0  497M   0% /run/secrets
tmpfs               497M     0  497M   0% /proc/kcore
```

图 24-22　Docker 容器扩容为 15GB

通过上述步骤成功地将 Docker 容器的 10GB 空间扩容为 15GB，还可以将上述步骤写成 shell 脚本，基于脚本参数快速扩容。给 Docker 磁盘扩容除了采用以上方法外，还可以使用挂载目录方法，基于-v 参数，在启动 Docker 容器时指定。

24.16　Docker 构建私有仓库

Docker 镜像默认存放在仓库中，Docker 仓库分为公共仓库和私有仓库，随着公司业务的发展，Docker 镜像的种类也非常繁多，为了统一管理，可以基于 registry 搭建本地私有

注意只是改变 20971520 的数字为 31457280，其他数字不变，通过命令 dmsetup resume 将修改后的容器文件激活，通过命令 dmsetup table 查看最新 Docker 扇区信息，如图 24-20 所示。

```
0 20971520 thin 253:0 30
[root@localhost ~]# echo 0 31457280 thin 253:0 30| dmsetup load docker-8:2-1704214-449c143b14b1
2e08807165602eff78ca2e39a5f444fb638b3d7131ccaa3d4843
[root@localhost ~]#
[root@localhost ~]# dmsetup resume docker-8:2-1704214-449c143b14b12e08807165602eff78ca2e39a5f44
4fb638b3d7131ccaa3d4843
[root@localhost ~]#
[root@localhost ~]# dmsetup table  docker-8:2-1704214-449c143b14b12e08807165602eff78ca2e39a5f44
4fb638b3d7131ccaa3d4843
0 31457280 thin 253:0 30
[root@localhost ~]#
```

图 24-20 Docker 容器扩容过程

（5）修改文件系统大小，命令为 resize2fs，如图 24-21 所示。

```
[root@localhost ~]# resize2fs /dev/mapper/docker-8:2-1704214-449c143b14b12e
39a5f444fb638b3d7131ccaa3d4843
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/mapper/docker-8:2-1704214-449c143b14b12e08807165602eff78
7131ccaa3d4843 is mounted on /var/lib/docker/devicemapper/mnt/449c143b14b12
e39a5f444fb638b3d7131ccaa3d4843; on-line resizing required
old_desc_blocks = 2, new_desc_blocks = 2
The filesystem on /dev/mapper/docker-8:2-1704214-449c143b14b12e08807165602e
8b3d7131ccaa3d4843 is now 3932160 blocks long.

[root@localhost ~]#
```

图 24-21 Docker 容器设备扩容

（6）验证 Docker rootfs 磁盘大小，如图 24-22 所示。

```
[root@localhost ~]# docker attach 449c143b14b1

[root@449c143b14b1 /]#
[root@449c143b14b1 /]#
[root@449c143b14b1 /]#
[root@449c143b14b1 /]# df -h
Filesystem            Size  Used Avail Use% Mounted on
rootfs                 15G  592M   14G   5% /
/dev/mapper/docker-8:2-1704214-449c143b14b12e08807165602eff78ca2e39a5
                       15G  592M   14G   5% /
tmpfs                 497M     0  497M   0% /dev
shm                    64M     0   64M   0% /dev/shm
tmpfs                 497M     0  497M   0% /run
tmpfs                 497M     0  497M   0% /tmp
/dev/sda2              29G  3.4G   24G  13% /etc/resolv.conf
/dev/sda2              29G  3.4G   24G  13% /etc/hostname
/dev/sda2              29G  3.4G   24G  13% /etc/hosts
tmpfs                 497M     0  497M   0% /run/secrets
tmpfs                 497M     0  497M   0% /proc/kcore
```

图 24-22 Docker 容器扩容为 15GB

通过上述步骤成功地将 Docker 容器的 10GB 空间扩容为 15GB，还可以将上述步骤写成 shell 脚本，基于脚本参数快速扩容。给 Docker 磁盘扩容除了采用以上方法外，还可以使用挂载目录方法，基于-v 参数，在启动 Docker 容器时指定。

24.16 Docker 构建私有仓库

Docker 镜像默认存放在仓库中，Docker 仓库分为公共仓库和私有仓库，随着公司业务的发展，Docker 镜像的种类也非常繁多，为了统一管理，可以基于 registry 搭建本地私有

仓库。

使用 Docker 私有仓库有以下优点：

- 节省网络带宽，针对每个镜像不用去 Docker 官网仓库下载；
- Docker 镜像从本地私有仓库中下载；
- 构建公司内部私有仓库，方便各部门使用，服务器管理更加统一；
- 可以基于 GIT 或 SVN、Jenkins 更新本地 Docker 私有仓库镜像版本。

官方提供 Docker registry 来构建本地私有仓库，目前最新版本为 v2，最新版的 Docker 已不再支持 v1，registry v2 使用 Go 语言编写，在性能和安全性上做了很多优化，重新设计了镜像的存储格式。以下为 Docker 本地私有仓库的构建方法及步骤。

(1) 下载 Docker registry 镜像，命令如下：

```
docker pull registry
```

(2) 启动私有仓库容器，启动命令如下：

```
mkdir -p /data/registry/
docker run -itd -p 5000:5000 -v /data/registry:/tmp/registry docker.io/registry
```

Docker 本地仓库启动后台容器，如图 24-23 所示。

```
[root@localhost ~]#
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREAT
docker.io/registry  latest              9d0c4eabab4d        5 wee
[root@localhost ~]#
[root@localhost ~]# mkdir -p  /data/registry/
[root@localhost ~]# docker run -itd  -p  5000:5000 -v /data/regist
53c8771c78524659dcc94b30c452da54273d02772e410566308cc2d234d94d19

[root@localhost ~]#
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND
    PORTS                   NAMES
53c8771c7852        docker.io/registry  "/entrypoint.sh /etc/"
    0.0.0.0:5000->5000/tcp  compassionate_mcnulty
[root@localhost ~]#
```

图 24-23　启动 Docker 仓库容器

默认情况下，会将仓库存放于容器内的/tmp/registry 目录下，这样如果容器被删除，则存放于容器中的镜像也会丢失，所以一般情况下会指定本地/data/registry 目录挂载到容器内的/tmp/registry 下。

(3) 上传镜像至本地私有仓库。客户端上传镜像至本地私有仓库，以 busybox 镜像为例，将 busybox 上传至私有仓库服务器，命令如下：

```
docker pull busybox
docker tag busybox 192.168.1.123:5000/busybox
docker push 192.168.1.123:5000/busybox
```

(4) 检测本地私有仓库，命令如下：

```
curl -XGET http://192.168.1.123:5000/v2/_catalog
curl -XGET http://192.168.1.123:5000/v2/busybox/tags/list
```

（5）客户端使用本地私有仓库。在客户端 Docker 配置文件/etc/sysconfig/docker 中添加以下代码，同时重启 Docker 服务，获取本地私有仓库，如图 24-24 所示。

```
OPTIONS = ' -- selinux - enabled  -- log - driver = journald  -- signature - verification = false
 -- insecure - registry 192.168.1.123:5000'
ADD_REGISTRY = ' -- add - registry 192.168.1.123:5000'
```

```
[root@localhost ~]# docker pull busybox
Using default tag: latest
Trying to pull repository 192.168.1.123:5000/busybox ...
latest: Pulling from 192.168.1.123:5000/busybox

27144aa8f1b9: Pull complete
Digest: sha256:be3c11fdba7cfe299214e46edc642e09514dbb9bbefcd0d
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# docker pull centos
Using default tag: latest
Trying to pull repository 192.168.1.123:5000/centos ...
latest: Pulling from 192.168.1.123:5000/centos

3927adf284d5: Pull complete
Digest: sha256:1e668815ede0326c2057fd17e472db04acdc6d62b480662
```

图 24-24 使用本地 Docker 私有仓库

至此，Docker 本地私有仓库部署完毕，可以向仓库中添加、更新 Docker 镜像，或查看、删除 Docker 仓库相关的镜像，操作命令如下：

```
curl  - XGET http://192.168.1.123:5000/v2/_catalog
curl  - XGET http://192.168.1.123:5000/v2/image_name/tags/list
curl  - X DELETE http://192.168.1.123:5000/v1/repositories/镜像名称
#v2 版本，官网不建议删除私有仓库中的镜像，可以基于 delete - docker - registry - image 工具
#删除 Docker 镜像
curl https://raw.githubusercontent.com/burnettk/delete - docker - registry - image/master/
delete_docker_registry_image.py | sudo tee /usr/local/bin/delete_docker_registry_image >/
dev/null
chmod a + x /usr/local/bin/delete_docker_registry_image
export REGISTRY_DATA_DIR = /data/registry/v2
delete_docker_registry_image  -- image centos:v1
```

24.17 Docker 自动化部署一

熟练使用手工方法创建 Docker 容器后，如果想批量应用于生产环境，需要编写能够实现自动安装并配置 Docker 虚拟化及桥接网络的脚本，同时使用 pipework 这个软件来配置容器 IP，能够实现容器简单的管理。以下为 CentOS 6.X Linux 系统一键安装、配置、管理 Docker 的 shell 脚本，脚本代码如下：

```
#!/bin/bash
#auto install docker and Create VM
#by jfedu.net 2017
#Define PATH Varablies
IPADDR = 'ifconfig |grep "Bcast"|awk '{print $2}'|cut  - d:  - f2|grep "192.168"|head  - 1'
```

```
GATEWAY='route -n|grep "UG"|awk '{print $2}'|grep "192.168"|head -1'
DOCKER_IPADDR=$1
IPADDR_NET='ifconfig |grep "Bcast"|awk '{print $2}'|cut -d: -f2|grep "192.168"|head -1|
awk -F. '{print $1"."$2"."$3"."xxx"}''
NETWORK=(
    HWADDR='ifconfig eth0 |egrep "HWaddr|Bcast" |tr "\n" " "|awk '{print $5,$7,$NF}'|sed -
e 's/addr://g' -e 's/Mask://g'|awk '{print $1}''
    IPADDR='ifconfig eth0 |egrep "HWaddr|Bcast" |tr "\n" " "|awk '{print $5,$7,$NF}'|sed -
e 's/addr://g' -e 's/Mask://g'|awk '{print $2}''
    NETMASK='ifconfig eth0 |egrep "HWaddr|Bcast" |tr "\n" " "|awk '{print $5,$7,$NF}'|sed
-e 's/addr://g' -e 's/Mask://g'|awk '{print $3}''
    GATEWAY='route -n|grep "UG"|awk '{print $2}''
)
if [ -z "$1" -o -z "$2" -o -z "$3" -o -z "$4" ];then
    echo -e "\033[32m----------------------------------\033[0m"
    echo -e "\033[32mPlease exec $0 IPADDR CPU(C) MEM(G) DISK(G),example $0 $IPADDR_NET 16
32 50\033[0m"
    exit 0
fi
CPU='expr $2 - 1'
if [ ! -e /usr/bin/bc ];then
    yum install bc -y >>/dev/null 2>&1
fi
MEM_F='echo $3 \* 1024|bc'
MEM='printf "%.0f\n" $MEM_F'
DISK=$4
USER=$5
REMARK=$6
ping $DOCKER_IPADDR -c 1 >>/dev/null 2>&1
if [ $? -eq 0 ];then

    echo -e "\033[32m----------------------------------\033[0m"
    echo -e "\033[32mThe IP address to be used,Please change other IP,exit.\033[0m"
    exit 0
fi
if [ ! -e /etc/init.d/docker ];then
    rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
    yum install docker-io -y
    yum install device-mapper* -y
    /etc/init.d/docker start
    if [ $? -ne 0 ];then
        echo "Docker install error ,please check."
        exit
    fi
fi
cd /etc/sysconfig/network-scripts/
    mkdir -p /data/backup/'date +%Y%m%d-%H%M'
```

```
    yes|cp ifcfg-eth* /data/backup/'date +%Y%m%d-%H%M'/
if
    [ -e /etc/sysconfig/network-scripts/ifcfg-br0 ];then
    echo
else
    cat > ifcfg-eth0 << EOF
    DEVICE=eth0
    BOOTPROTO=none
    ${NETWORK[0]}
    NM_CONTROLLED=no
    ONBOOT=yes
    TYPE=Ethernet
    BRIDGE="br0"
    ${NETWORK[1]}
    ${NETWORK[2]}
    ${NETWORK[3]}
    USERCTL=no
EOF
    cat > ifcfg-br0 << EOF
    DEVICE="br0"
    BOOTPROTO=none
    ${NETWORK[0]}
    IPV6INIT=no
    NM_CONTROLLED=no
    ONBOOT=yes
    TYPE="Bridge"
    ${NETWORK[1]}
    ${NETWORK[2]}
    ${NETWORK[3]}
    USERCTL=no
EOF

    /etc/init.d/network restart

fi
echo 'Your can restart Ethernet Service: /etc/init.d/network restart !'
echo '-----------------------------------------------------------'
cd -
#######create docker container
service docker status >>/dev/null
if [ $? -ne 0 ];then
    /etc/init.d/docker restart
fi
NAME="Docker$$_'echo $DOCKER_IPADDR|awk -F"." '{print $(NF-1)"_"$NF}''"
IMAGES='docker images|grep -v "REPOSITORY"|grep -v "none"|head -1|awk '{print $1}''
CID=$(docker run -itd --cpuset-cpus=0-$CPU -m ${MEM}m --net=none --name=$NAME
 $IMAGES /bin/bash)
```

```
if [ -z $IMAGES ];then
    echo "Plesae Download Docker CentOS Images, you can to be use docker search CentOS, and
docker pull CentOS6.5-ssh,exit 0"
    exit 0
fi

if [ ! -f /usr/local/bin/pipework ];then
    yum install wget unzip zip -y
    wget https://github.com/jpetazzo/pipework/archive/master.zip
        unzip master
        cp pipework-master/pipework /usr/local/bin/
        chmod +x /usr/local/bin/pipework
        rm -rf master
fi

ip netns >>/dev/null
if [ $? -ne 0 ];then
    rpm -e iproute --nodeps
        rpm -ivh https://repos.fedorapeople.org/openstack/EOL/openstack-grizzly/epel-6/
iproute-2.6.32-130.el6ost.netns.2.x86_64.rpm
fi
pipework br0 $NAME $DOCKER_IPADDR/24@$IPADDR

docker ps -a |grep "$NAME"

DEV=$(basename $(echo /dev/mapper/docker-*-$CID))
dmsetup table $DEV | sed "s/0 [0-9]* thin/0 $(( ${DISK} * 1024 * 1024 * 1024/512)) thin/" |
dmsetup load $DEV
dmsetup resume $DEV
resize2fs /dev/mapper/$DEV
docker start $CID
docker logs $CID
LIST="docker_vmlist.csv"
if [ ! -e $LIST ];then
    echo "编号,容器ID,容器名称,CPU,内存,硬盘,容器IP,宿主机IP,使用人,备注" >$LIST
fi
###################
NUM='cat docker_vmlist.csv |grep -v CPU|tail -1|awk -F, '{print $1}''
if [[ $NUM -eq "" ]];then
        NUM="1"
else
        NUM='expr $NUM + 1'
fi
##################
echo -e "\033[32mCreate virtual client Successfully.\n$NUM 'echo $CID|cut -b 1-12' $NAME
 $2C ${MEM}M ${DISK}G $DOCKER_IPADDR $IPADDR $USER $REMARK\033[0m"
```

```
if [ -z $USER ];then
    USER="NULL"
    REMARK="NULL"
fi
echo $NUM,'echo $CID|cut -b 1-12', $NAME, ${2}C, ${MEM}M, ${DISK}G, $DOCKER_IPADDR,
 $IPADDR, $USER, $REMARK >> $LIST
rm -rf docker_vmlist_*
iconv -c -f utf-8 -t gb2312 docker_vmlist.csv -o docker_vmlist_'date +%H%M'.csv
```

24.18 Docker 自动化部署二

随着 Linux 技术的发展，目前越来越多的企业开始使用 CentOS 7 系统，以下为 CentOS 7.X Linux 系统一键安装、配置、管理 Docker 的 shell 脚本，脚本代码如下：

```
#!/bin/bash
#auto install docker and Create VM
#by jfedu.net 2017
#Define PATH Varablies
IPADDR='ifconfig|grep -E "\<inet\>"|awk '{print $2}'|grep "192.168"|head -1'
GATEWAY='route -n|grep "UG"|awk '{print $2}'|grep "192.168"|head -1'
IPADDR_NET='ifconfig|grep -E "\<inet\>"|awk '{print $2}'|grep "192.168"|head -1|awk -F.
'{print $1"."$2"."$3"."}''
LIST="/root/docker_vmlist.csv"
if [ ! -f /usr/sbin/ifconfig ];then
    yum install net-tools* -y
fi
for i in 'seq 1 253';do ping -c 1 ${IPADDR_NET}${i} ;[ $? -ne 0 ]&& DOCKER_IPADDR=
"${IPADDR_NET}${i}" &&break;done >>/dev/null 2>&1
echo "##################"
echo -e "Dynamic get docker IP,The Docker IP address\n\n$DOCKER_IPADDR"
NETWORK=(
    HWADDR='ifconfig eth0|grep ether|awk '{print $2}''
    IPADDR='ifconfig eth0|grep -E "\<inet\>"|awk '{print $2}''
    NETMASK='ifconfig eth0|grep -E "\<inet\>"|awk '{print $4}''
    GATEWAY='route -n|grep "UG"|awk '{print $2}''
)
if [ -z "$1" -o -z "$2" ];then

    echo -e "\033[32m-----------------------------------\033[0m"
    echo -e "\033[32mPlease exec $0 CPU(C) MEM(G),example $0 4 8\033[0m"
    exit 0
fi
#CPU='expr $2 - 1'
if [ ! -e /usr/bin/bc ];then
    yum install bc -y >>/dev/null 2>&1
```

```
fi
CPU_ALL='cat /proc/cpuinfo |grep processor|wc -l'
if [ ! -f $LIST ];then
    CPU_COUNT=$1
    CPU_1="0"
    CPU1='expr $CPU_1 + 0'
    CPU2='expr $CPU1 + $CPU_COUNT - 1'
    if [ $CPU2 -gt $CPU_ALL ];then
        echo -e "\033[32mThe System CPU count is $CPU_ALL,not more than it.\033[0m"
        exit
    fi
else
    CPU_COUNT=$1
    CPU_1='cat $LIST|tail -1|awk -F"," '{print $4}'|awk -F"-" '{print $2}''
    CPU1='expr $CPU_1 + 1'
    CPU2='expr $CPU1 + $CPU_COUNT - 1'
    if [ $CPU2 -gt $CPU_ALL ];then
        echo -e "\033[32mThe System CPU count is $CPU_ALL,not more than it.\033[0m"
        exit
    fi
fi
MEM_F='echo $2 \* 1024|bc'
MEM='printf "%.0f\n" $MEM_F'
DISK=20
USER=$3
REMARK=$4
ping $DOCKER_IPADDR -c 1 >>/dev/null 2>&1

if [ $? -eq 0 ];then

    echo -e "\033[32m-----------------------------------\033[0m"
    echo -e "\033[32mThe IP address to be used,Please change other IP,exit.\033[0m"
    exit 0
fi

if [ ! -e /usr/bin/docker ];then
    yum install docker* device-mapper* lxc -y
    mkdir -p /export/docker/
    cd /var/lib/ ;rm -rf docker ;ln -s /export/docker/ .
    mkdir -p /var/lib/docker/devicemapper/devicemapper
    dd if=/dev/zero of=/var/lib/docker/devicemapper/devicemapper/data bs=1G count=0
    seek=2000
    service docker start
    if [ $? -ne 0 ];then
        echo "Docker install error ,please check."
        exit
    fi
```

```
fi

cd /etc/sysconfig/network-scripts/
    mkdir -p /data/backup/'date +%Y%m%d-%H%M'
    yes|cp ifcfg-eth* /data/backup/'date +%Y%m%d-%H%M'/
if
    [ -e /etc/sysconfig/network-scripts/ifcfg-br0 ];then
    echo
else
    cat > ifcfg-eth0 << EOF
    DEVICE=eth0
    BOOTPROTO=none
     ${NETWORK[0]}
    NM_CONTROLLED=no
    ONBOOT=yes
    TYPE=Ethernet
    BRIDGE="br0"
     ${NETWORK[1]}
     ${NETWORK[2]}
     ${NETWORK[3]}
    USERCTL=no
EOF
    cat > ifcfg-br0 << EOF
    DEVICE="br0"
    BOOTPROTO=none
     ${NETWORK[0]}
    IPV6INIT=no
    NM_CONTROLLED=no
    ONBOOT=yes
    TYPE="Bridge"
     ${NETWORK[1]}
     ${NETWORK[2]}
     ${NETWORK[3]}
    USERCTL=no
EOF
    /etc/init.d/network restart

fi
echo 'Your can restart Ethernet Service: /etc/init.d/network restart !'
echo '--------------------------------------------------------'

cd -
#######create docker container
service docker status >>/dev/null
if [ $? -ne 0 ];then
    service docker restart
fi
```

```
NAME = "Docker_'echo $DOCKER_IPADDR|awk - F"." '{print $(NF - 1)"_" $NF}''"
IMAGES = 'docker images|grep - v "REPOSITORY"|grep - v "none"|grep "CentOS"|head - 1|awk '
{print $1}''
if [ - z $IMAGES ];then
    echo "Plesae Download Docker CentOS Images, you can to be use docker search CentOS, and
docker pull CentOS6.5 - ssh,exit 0"
    if [ ! - f jfedu_CentOS68.tar ];then
        echo "Please upload jfedu_CentOS68.tar for docker server."
        exit
    fi
    cat jfedu_CentOS68.tar|docker import - jfedu_CentOS6.8
fi
IMAGES = 'docker images|grep - v "REPOSITORY"|grep - v "none"|grep "CentOS"|head - 1|awk
'{print $1}''
CID = $(docker run - itd -- privileged -- cpuset - cpus = ${CPU1} - ${CPU2} - m ${MEM}m --
net = none -- name = $NAME $IMAGES /bin/bash)
echo $CID
docker ps - a |grep "$NAME"
pipework br0 $NAME $DOCKER_IPADDR/24@ $IPADDR
docker exec $NAME /etc/init.d/sshd start
if [ ! - e $LIST ];then
    echo "编号,容器 ID,容器名称,CPU,内存,硬盘,容器 IP,宿主机 IP,使用人,备注" > $LIST
fi
###################
NUM = 'cat $LIST |grep - v CPU|tail - 1|awk - F, '{print $1}''
if [[ $NUM - eq "" ]];then
        NUM = "1"
else
        NUM = 'expr $NUM + 1'
fi
##################
echo - e "\033[32mCreate virtual client Successfully.\n $NUM 'echo $CID|cut - b 1 - 12', $NAME,
$CPU1 - $CPU2, ${MEM}M, ${DISK}G, $DOCKER_IPADDR, $IPADDR, $USER, $REMARK\033[0m"
if [ - z $USER ];then
    USER = "NULL"
    REMARK = "NULL"
fi
echo $NUM,'echo $CID|cut - b 1 - 12', $NAME, $CPU1 - $CPU2, ${MEM}M, ${DISK}G, $DOCKER_IPADDR,
$IPADDR, $USER, $REMARK >> $LIST
rm - rf /root/docker_vmlist_ *
iconv - c - f utf - 8 - t gb2312 $LIST - o /root/docker_vmlist_'date + %H%M'.csv
```

第 25 章 Openstack+KVM 构建企业私有云



随着互联网技术的变革，云计算技术被越来越多的企业使用，包括京东、百度、阿里巴巴、腾讯等互联网企业，其中 Openstack 项目目标是提供实施简单、可大规模扩展、丰富、标准统一的云计算管理平台。Openstack 通过各种互补的服务提供了基础设施即服务(infrastructure as a service，IaaS)的解决方案，每个服务提供 API 以进行集成。

本章向读者介绍 Openstack 云计算入门、Openstack 简介、Openstack 各个组件功能、Openstack 各个组件安装、MQ 消息队列及应用案例、Openstack 故障排错、构建 Openstack 私有云平台、虚拟机管理、镜像导入、创建安全策略等内容。

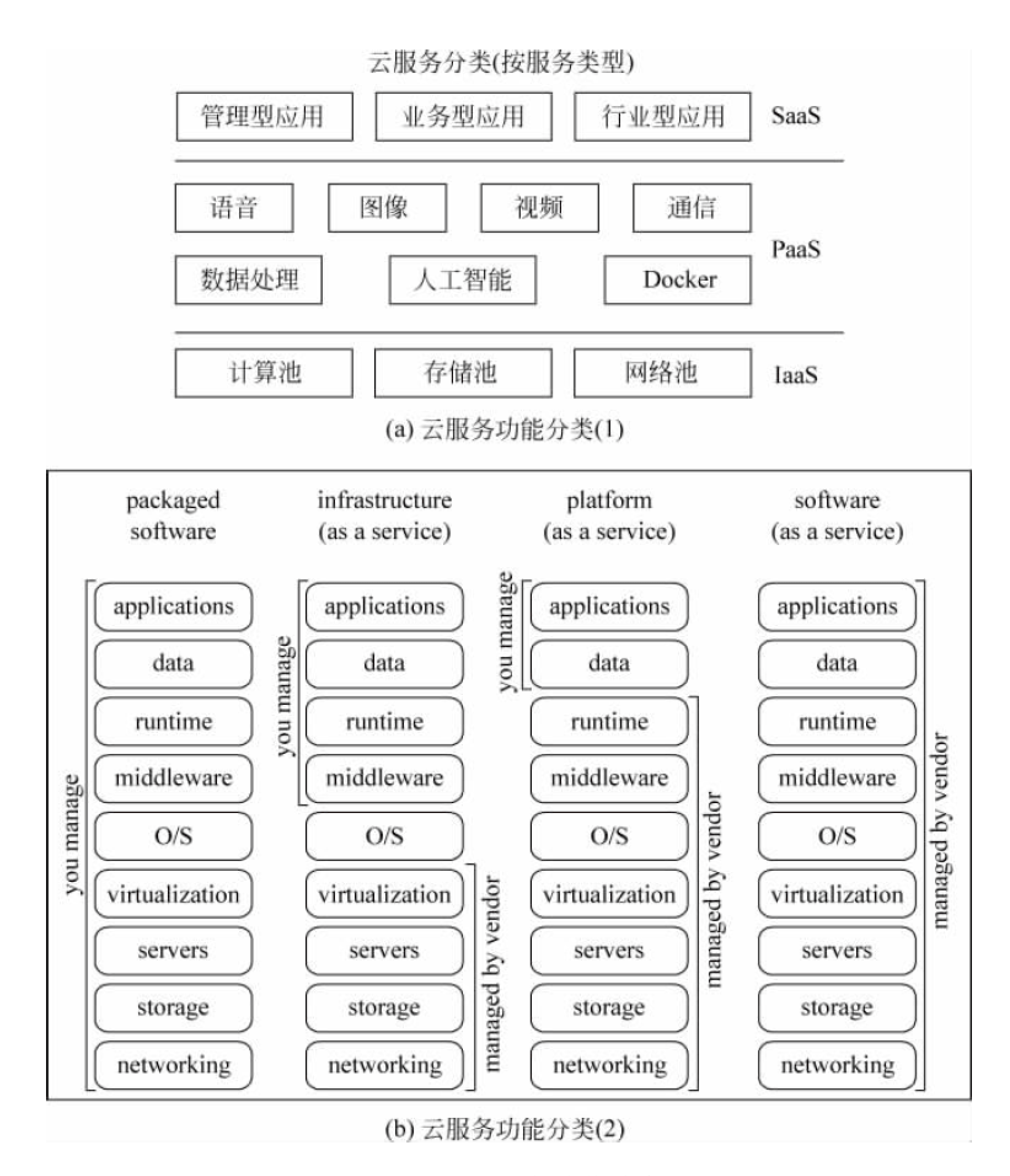
25.1 云计算及 Openstack 入门

云计算(cloud computing)是基于互联网相关服务资源的增加、使用和交付为主的一体化解决方案，通过互联网来提供动态易扩展的虚拟化的资源。

对于云计算的概念理解有上百种说法，而美国国家标准与技术研究院(NIST)定义为云计算是一种按使用量付费的模式。这种模式提供可用的、便捷的、按需的网络访问，进入可配置的计算资源共享池，计算资源包括网络、服务器、存储、应用软件、服务等，这些资源能够被快速提供，且只需投入很少的管理工作，或与服务供应商进行很少的交互。

而 Openstack 是一个由美国国家航空航天局(National Aeronautics and Space Administration，NASA)和 Rackspace 合作研发并发起，以 Apache 许可证授权的自由软件和开放源代码项目，是一个开源的云计算管理平台项目，由几个主要的组件组合起来完成具体工作。

Openstack 支持几乎所有类型的云环境，项目目标是提供实施简单、可大规模扩展、丰富、标准统一的云计算管理平台。Openstack 通过各种互补的服务提供了基础设施即服务(infrastructure as a service，IaaS)的解决方案，每个服务提供 API 以进行集成，当然除了 IaaS 解决方案，还有主流的平台即服务(platform-as-a-service，PaaS)和软件即服务(software-as-a-service，SaaS)，IaaS、PaaS、SaaS 三种云计算服务的区别如图 25-1 所示。



(a) 云服务功能分类(1)

(b) 云服务功能分类(2)

图 25-1 云服务功能分类

Openstack 是一个旨在为公共及私有云的建设与管理提供软件的开源项目，它的社区拥有超过 130 家企业及 1350 位开发者，这些机构与个人都将 Openstack 作为基础设施即服务(IaaS)资源的通用前端。Openstack 项目的首要任务是简化云的部署过程并为其带来良好的可扩展性。

Openstack 云计算平台，帮助服务商和企业内部实现类似于 Amazon EC2 和 S3 的云基础架构服务。Openstack 包含两个主要模块，即 Nova 和 Swift，前者是 NASA 开发的虚拟服务器部署和业务计算模块，后者是 Rackspace 开发的分布式云存储模块，两者可以一起用，也可以分开单独用。

Openstack 除了有 Rackspace 和 NASA 的大力支持外，还有包括 Dell、Citrix、Cisco、Canonical 等重量级公司的贡献和支持，发展速度非常快，有取代另一个业界领先开源云平台 Eucalyptus 的态势。

Openstack 遵循一年两次的开发及发布的周期，在春末提供一个发布，秋季提供第二个版本，使用版本的代号按 A～Z 字母顺序排列，目前 Mitika 版本是最新稳定版本，如图 25-2 所示。

series	status	initial release date	next phase	EOL date
Queens	Future	proposed		TBD
Pike	Under Development	scheduled		TBD
Ocata	Phase I – Latest release	2017-02-22	Phase II – Maintained release on 2017-08-28	2018-02-26
Newton	Phase II – Maintained release	2016-10-06	Phase III – Legacy release on 2017-10-09	2017-10-11
Mitaka	EOL	2016-04-07		2017-04-10
Liberty	EOL	2015-10-15		2016-11-17
Kilo	EOL	2015-04-30		2016-05-02
Juno	EOL	2014-10-16		2015-12-07
Icehouse	EOL	2014-04-17		2015-07-02
Havana	EOL	2013-10-17		2014-09-30
Grizzly	EOL	2013-04-04		2014-03-29
Folsom	EOL	2012-09-27		2013-11-19
Essex	EOL	2012-04-05		2013-05-06
Diablo	EOL	2011-09-22		2013-05-06
Cactus	Deprecated	2011-04-15		
Bexar	Deprecated	2011-02-03		
Austin	Deprecated	2010-10-21		

图 25-2　Openstack 发展版本

25.2　Opentstack 核心组件

Openstack 覆盖了网络、虚拟化、操作系统、服务器等各个方面。它是一个正在开发中的云计算平台项目，根据成熟及重要程度的不同，被分解成核心项目、孵化项目，以及支持项目和相关项目。

每个项目都有自己的委员会和项目技术主管，而且每个项目都不是一成不变的，孵化项目可以根据发展的成熟度和重要性，转变为核心项目。截止到 Icehouse 版本，以下为 Openstack 依赖的 10 个核心项目，主要组件功能和调用关系如图 25-3 所示。

- 计算(compute)：Nova，一套控制器，用于为单个用户或使用群组管理虚拟机实例的整个生命周期，根据用户需求来提供虚拟服务。负责虚拟机创建、开机、关机、挂起、暂停、调整、迁移、重启、销毁等操作，配置 CPU、内存等信息规格。
- 对象存储(object storage)：Swift，一套用于在大规模可扩展系统中通过内置冗余及高容错机制实现对象存储的系统，允许进行存储或检索文件，可为 Glance 提供镜像存储，为 Cinder 提供卷备份服务。
- 镜像服务(image service)：Glance，一套虚拟机镜像查找及检索系统，支持多种虚拟机镜像格式(AKI、AMI、ARI、ISO、QCOW2、Raw、VDI、VHD、VMDK)，有创建上传

Openstack的主要功能组件

App Dev　Dev Ops　App Owner　Cloud Ops

Horizon——UI服务
Keystone——身份服务
Nova——计算服务
Glance——镜像服务
Cinder/Swift——存储服务
Neutron——网络服务
Heat——编排组织服务
Ceilometer——监控计量服务

billing　identity　integration

compute API　image API　user dashboard　customer portal　Presentation

orchos-tration　schedu-ling　policy　image registry　logging　logic(control)

compute　volume　network　resources

admin API　monito-ring　manage-ment

(a) Openstack组件功能

Horizon　dashboard

Quantum　network

Nova　compute

Glance　image

Swift　object storage

block storage　Cinder

Keystone　identity

用户界面　用户界面　用户界面　提供认证　用户界面　用户界面

提供网络连接　保存镜像文件　保存磁盘文件

提供外部磁盘卷　提供认证　提供认证　提供认证　提供认证　提供认证

(b) Openstack组件调用关系

图 25-3　Openstack 主要组件功能和调用关系

镜像、删除镜像、编辑镜像基本信息的功能。

- 身份服务(identity service)：Keystone，为 Openstack 其他服务提供身份验证、服务规则和服务令牌的功能，管理 domains、projects、users、groups、roles。
- 网络 & 地址管理(network)：Neutron，提供云计算的网络虚拟化技术，为 Openstack 其他服务提供网络连接服务。为用户提供接口，可以定义 network、subnet、router、DHCP、DNS、负载均衡、L3 服务、GRE、VLAN 等。插件架构支持许多主流的网络厂家和技术，如 Openvswitch。
- 块存储(block storage)：Cinder，为运行实例提供稳定的数据块存储服务，它的插件驱动架构有利于块设备的创建和管理，如创建卷、删除卷，在实例上挂载和卸载卷。
- UI 界面(dashboard)：Horizon，Openstack 中各种服务的 Web 管理门户，用于简化用户对服务的操作，例如启动实例、分配 IP 地址、配置访问控制等。
- 测量(metering)：Ceilometer，像一个漏斗一样，能把 Openstack 内部发生的几乎所有的事件都收集起来，然后为计费和监控以及其他服务提供数据支撑。
- 部署编排(orchestration)：Heat，提供了一种通过模板定义的协同部署方式，实现云基础设施软件运行环境(计算、存储和网络资源)的自动化部署。
- 数据库服务(database service)：Trove，为用户在 Openstack 的环境提供可扩展和可靠的关系以及非关系数据库引擎服务。

25.3 Openstack 准备环境

构建完整的 Openstack 私有云平台，生产环境至少需要 2 台服务器，一台为控制节点服务器，一台为计算节点服务器，不推荐把计算节点服务安装在控制节点服务器，宿主机操作系统推荐使用 CentOS 7.X 系列，以下为构建 Openstack 基础环境信息：

操作系统版本：CentOS Linux release 7.3.1611

192.168.1.120 node 1 控制节点

192.168.1.121 node 2 计算节点

Openstack 官方提示线上生产环境 Openstack 各个节点的硬件配置，如图 25-4 所示。

其中 node 1 控制节点主要用于操控计算节点，node 2 计算节点为创建虚拟机的资源池，node 1 控制节点主要配置服务包括：MySQL、RabbitMQ、Apache、Horizon、Keystone、Glance、Nova(API、Cert、Scheduler、ConsoleAuth、Conductor、NoVNCporxy)、Neutron(server、LinuxBridge-Agent)、Cinder(API、Scheduler、Volume 及可选 GFS 分布式存储)等，如图 25-5 所示。

node 2 计算节点主要配置服务包括：Nova(Nova-Compute、Libvirt、KVM)、Neutron(LinuxBridge-Agent)等，如图 25-6 所示。

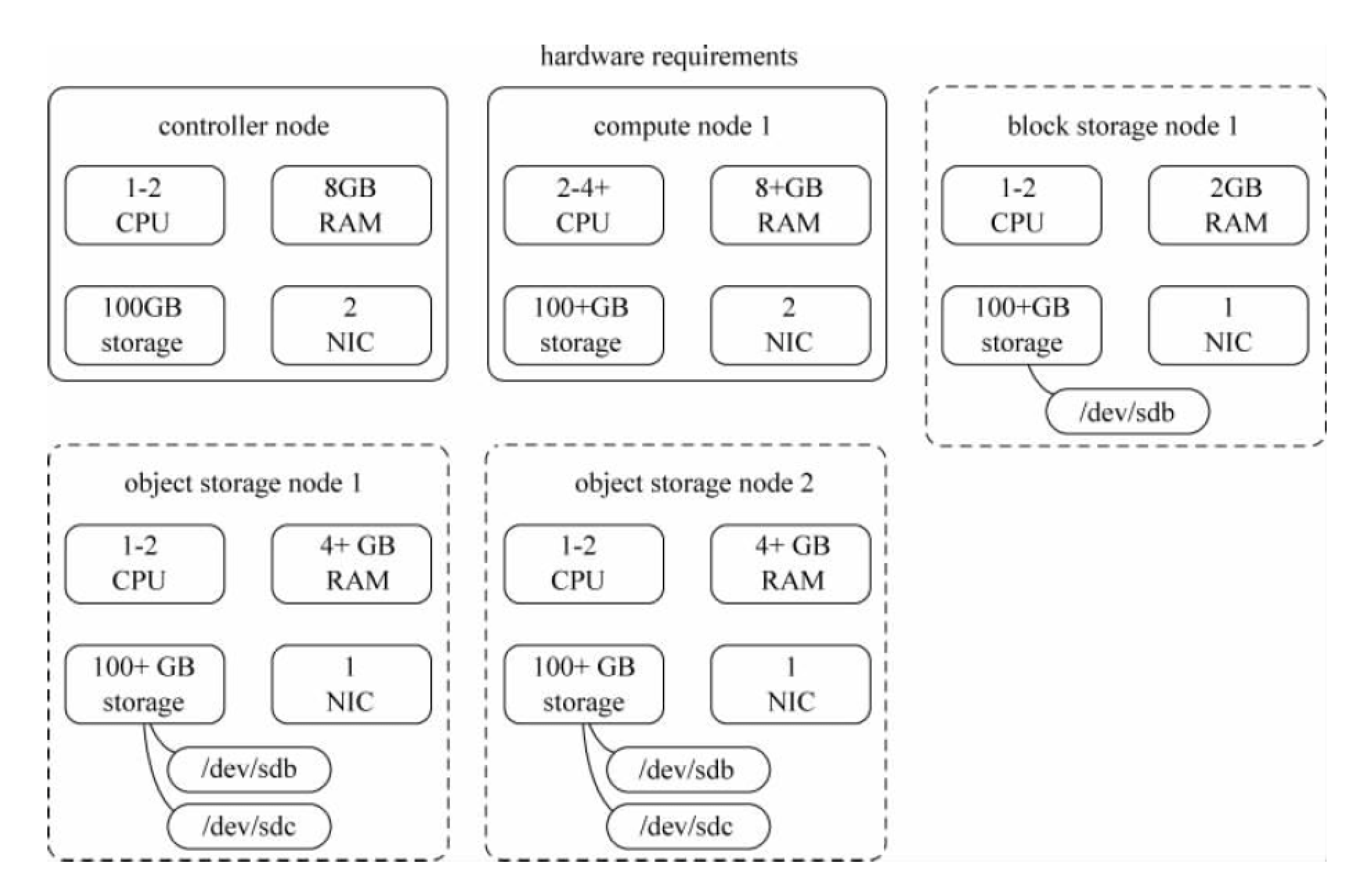


图 25-4　Openstack 生产环境硬件要求

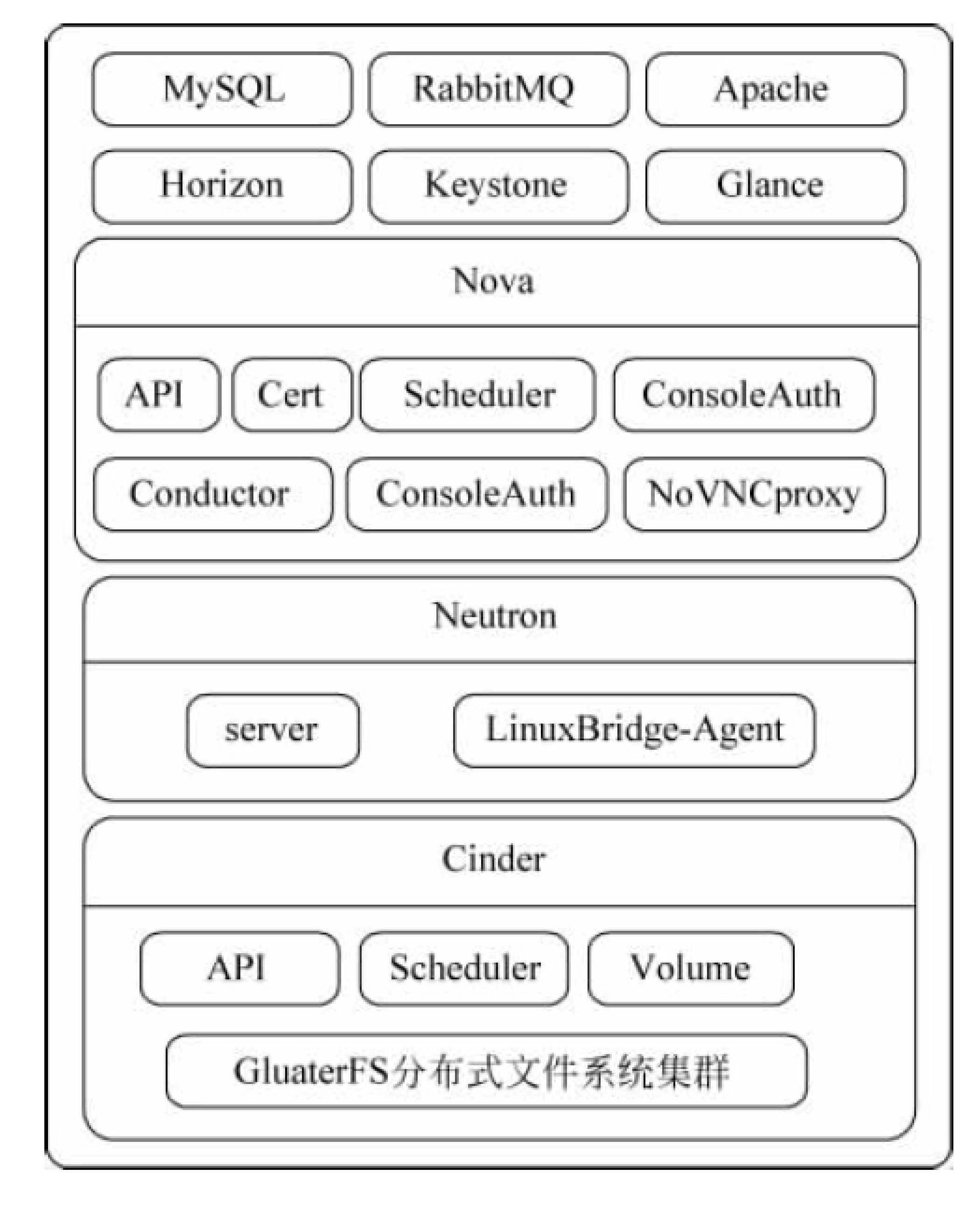


图 25-5　Openstack 控制节点组件

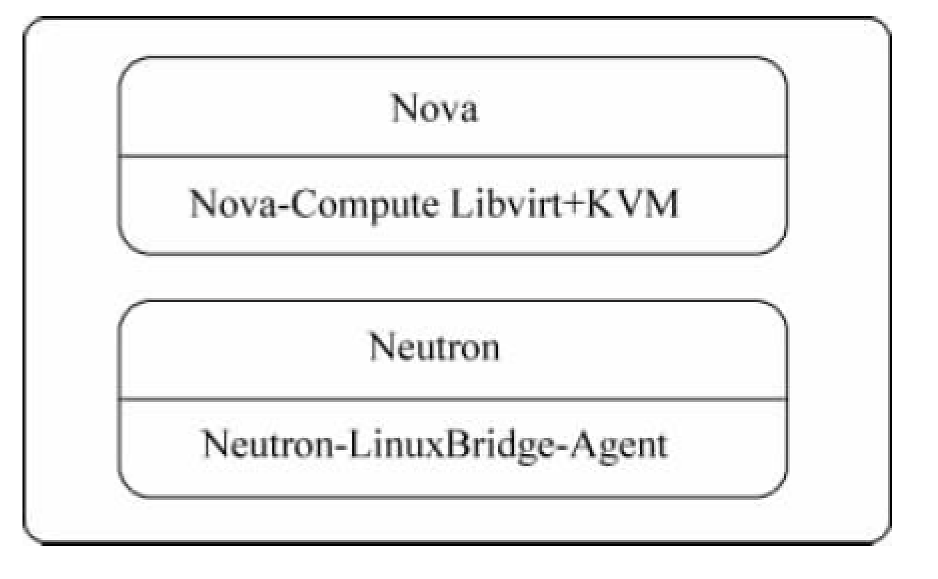


图 25-6　Openstack 计算节点组件

25.4 主机名及防火墙设置

构建 Openstack 平台，由于节点之间相互通信会解析本地主机名，需提前设置主机名，生产环境可以基于 DNS 解析，控制节点 node 1 和计算节点 node 2 上进行以下配置，配置本地主机名及关闭防火墙、关闭 SELinux 服务，并且基于 ntpdate 工具同步各个节点的时间，配置代码如下：

```
cat >/etc/hosts << EOF
127.0.0.1       localhost localhost.localdomain
#103.27.60.52   mirror.CentOS.org
#66.241.106.180 mirror.CentOS.org
192.168.1.120   node1
192.168.1.121   node2
EOF
#永久关闭 SELinux 服务
sed -i '/SELINUX/s/enforcing/disabled/g' /etc/sysconfig/selinux
#临时关闭 SELinux 服务
setenforce  0
#停止防火墙、禁止开机启动
systemctl stop firewalld.service
systemctl disable firewalld.service
#同步服务器时间
ntpdate pool.ntp.org
hostname 'cat /etc/hosts|grep $(ifconfig|grep broadcast|awk '{print $2}')|awk '{print $2}'';su
```

25.5 Openstack 服务安装

Openstack node 1 主控制节点作为 Openstack 集群服务器端，主要用于对整个 Openstack 私有云控制与调度，node 1 节点必备服务安装配置如下：

```
#清理 Openstack liberty 版本 YUM 元数据
yum --enablerepo=CentOS-openstack-liberty clean metadata
#安装 epel 扩展源及 Openstack liberty YUM 源
yum install -y epel-release
yum install -y CentOS-release-openstack-liberty
yum install -y python-openstackclient
#安装 MariaDB 数据库及 Python 数据库模块支持
yum install -y mariadb mariadb-server MySQL-python mariadb-devel
#安装 MQ 消息队列服务 RabbitMQ
yum install -y rabbitmq-server
#安装认证中心 Keystone、HTTP Web 服务、Memacached 缓存服务
yum install -y openstack-keystone httpd httpd-devel mod_wsgi memcached python-memcached
#安装镜像管理 Glance 服务
```

```
yum install -y openstack-glance python-glance python-glanceclient
#安装 Openstack 管理模块、虚拟机控制调度及控制服务 Nova
yum install -y openstack-nova-api openstack-nova-cert openstack-nova-conductor
openstack-nova-console openstack-nova-novncproxy openstack-nova-scheduler python-
novaclient
#安装 Neutron 网络配置及管理服务
yum install -y openstack-neutron openstack-neutron-ml2 openstack-neutron-linuxbridge
python-neutronclient ebtables ipset
#安装 Openstack 控制界面 dashboard
yum install -y openstack-dashboard
#安装镜像块存储服务 Cinder
yum install -y openstack-cinder python-cinderclient
#升级 KVM 虚拟机管理程序 Qemu 服务
yum install -y CentOS-release-qemu-ev.noarch
yum -y install qemu-kvm qemu-img
#调整 MariaDB 最大连接数为 2000
sed -i '/\[mysqld\]/amax_connections=2000' /etc/my.cnf
#将 MariaDB 服务设置开机启动,并且启动 MariaDB 服务
systemctl enable mariadb.service
systemctl start mariadb.service
```

node 1 节点创建数据库配置如下：

```
#进入 MariaDB 数据库,创建必备库,同时授权本机和计算节点能访问
mysql
create database keystone
grant all on keystone.* to 'keystone'@'localhost' identified by 'keystone'
grant all on keystone.* to 'keystone'@'%' identified by 'keystone'
grant all on keystone.* to 'keystone'@'node1' identified by 'keystone'
create database glance
grant all on glance.* to 'glance'@'localhost' identified by 'glance'
grant all on glance.* to 'glance'@'%' identified by 'glance'
grant all on glance.* to 'glance'@'node1' identified by 'glance'
create database nova
grant all on nova.* to 'nova'@'localhost' identified by 'nova'
grant all on nova.* to 'nova'@'%' identified by 'nova'
grant all on nova.* to 'nova'@'node1' identified by 'nova'
create database neutron
grant all on neutron.* to 'neutron'@'localhost' identified by 'neutron'
grant all on neutron.* to 'neutron'@'%' identified by 'neutron'
grant all on neutron.* to 'neutron'@'node1' identified by 'neutron'
create database cinder
grant all on cinder.* to 'cinder'@'localhost' identified by 'cinder'
grant all on cinder.* to 'cinder'@'%' identified by 'cinder'
grant all on cinder.* to 'cinder'@'node1' identified by 'cinder'
flush privileges
exit
```

25.6　MQ 消息队列服务

MQ 全称为 message queue，即消息队列。MQ 是一种应用程序对应用程序的通信方法，应用程序通过读写出入队列的消息(针对应用程序的数据)来通信，而无须专用连接来链接它们。

25.6.1　MQ 消息队列简介

消息队列中间件是分布式系统中重要的组件，主要解决应用耦合、异步消息、流量削锋等问题，实现高性能、高可用、可伸缩和最终一致性架构。主流消息队列包括 ActiveMQ、RabbitMQ、ZeroMQ、Kafka、MetaMQ、RocketMQ 等。

消息传递指的是程序之间通过在消息中发送数据进行通信，而不是通过直接调用彼此来通信，直接调用通常是用于诸如远程过程调用的技术。

排队指的是应用程序通过队列来通信，队列的使用除去了接收和发送应用程序同时执行的要求。RabbitMQ 是一个在 AMQP 基础上完整的、可复用的企业消息系统，遵循 GPL 开源协议。

Openstack 的架构决定了需要使用消息队列机制来实现不同模块间的通信，通过消息验证、消息转换、消息路由架构模式，带来的好处就是可以使模块之间最大程度解耦，客户端不需要关注服务端的位置和是否存在，只需通过消息队列进行信息的发送。

RabbitMQ 适合部署在一个拓扑灵活易扩展的规模化系统环境中，有效保证不同模块、不同节点、不同进程之间消息通信的时效性，可有效支持 Openstack 云平台系统的规模化部署、弹性扩展、灵活架构以及信息安全的需求。

25.6.2　RabbitMQ 应用场景

随着互联网 IT 技术发展，MQ 应用场景非常广泛，为了能满足大规模用户访问，对网站性能也要求越来越严格，如果企业网站各个系统相互紧密依赖，要是一个小系统崩溃，就会影响巨大，所以使用 MQ 消息队列可以让各个系统更加可靠、稳定地对外提供服务，以下为 MQ 消息队列应用场景。

1. MQ 异步信息场景

MQ 应用的场合非常的多，例如某个 Discuz 论坛网站，用户注册信息后，需要发注册邮件和注册短信然后才能确认用户注册成功，如图 25-7 所示为没有使用 MQ 消息通信的服务架构。

Web 网站将注册信息写入 MySQL 数据库成功后，发送注册邮件，再发送注册短信。以上三个任务全部完成后，返回给客户端，总共花费时间为 150ms。

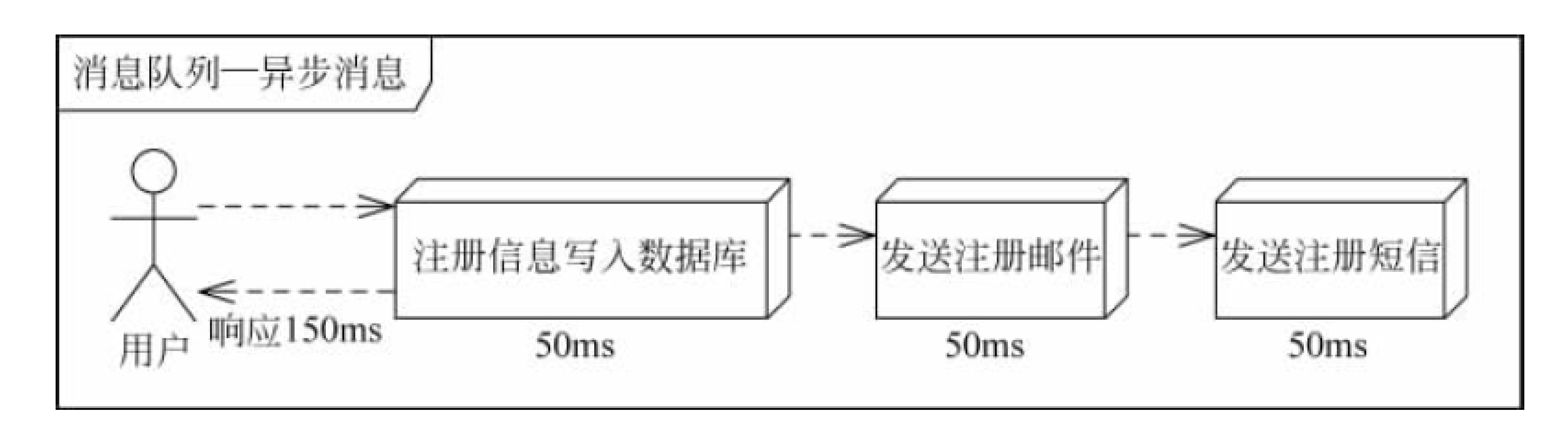


图 25-7　传统网站底层服务调用架构

而引入 MQ 消息队列后，将原来的网站业务逻辑进行调整，采用异步处理，改造后的架构如图 25-8 所示。

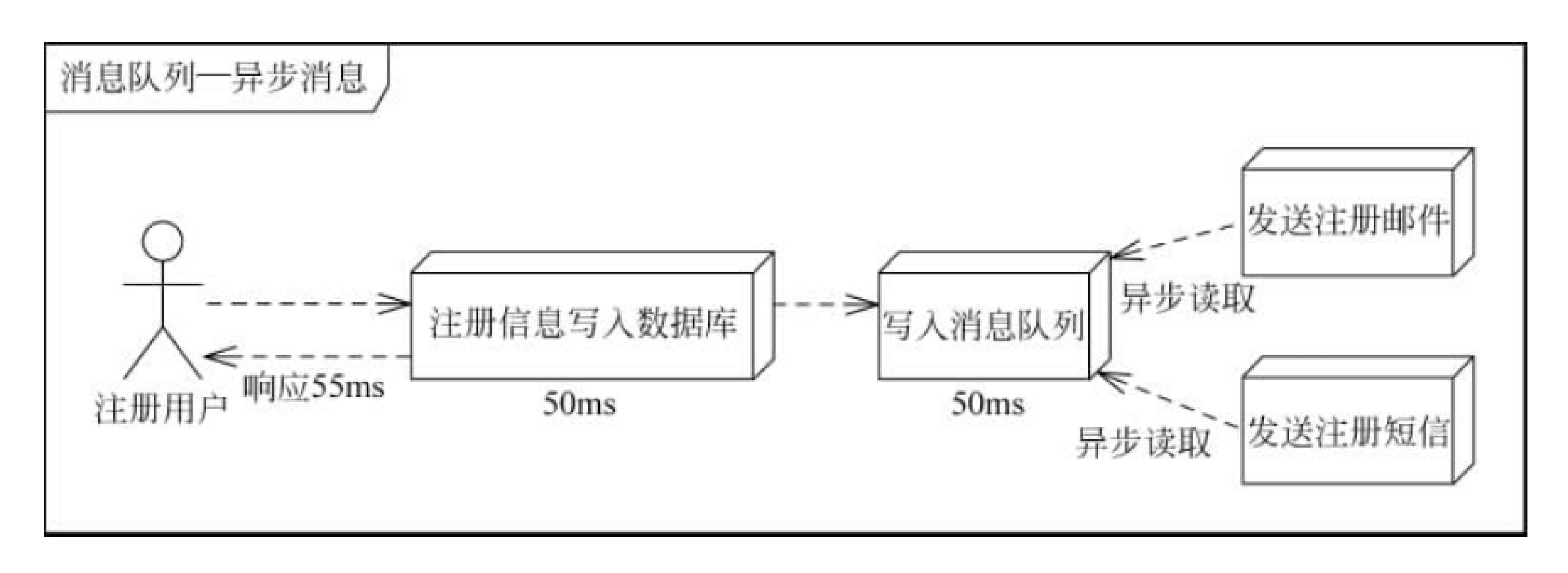


图 25-8　引入 MQ 消息队列服务架构

论坛注册用户的响应时间是注册信息写入数据库的时间 50ms，将注册用户的消息写入到 MQ 队列，然后注册邮件和注册短信的模块，直接来 MQ 队列读取消息，从而节省用户注册时间，提高网站对外体验，因此用户的响应时间可能约等于 50ms。

2. MQ 应用解耦场景

某购物网站，用户下单后购买产品，下单成功后订单系统需要通知库存系统，然后库存系统的库存数减一。传统的网站服务架构为订单系统调用库存系统的接口，缺点是假如库存系统无法访问，则订单减库存将失败，从而导致订单失败，如图 25-9 所示。

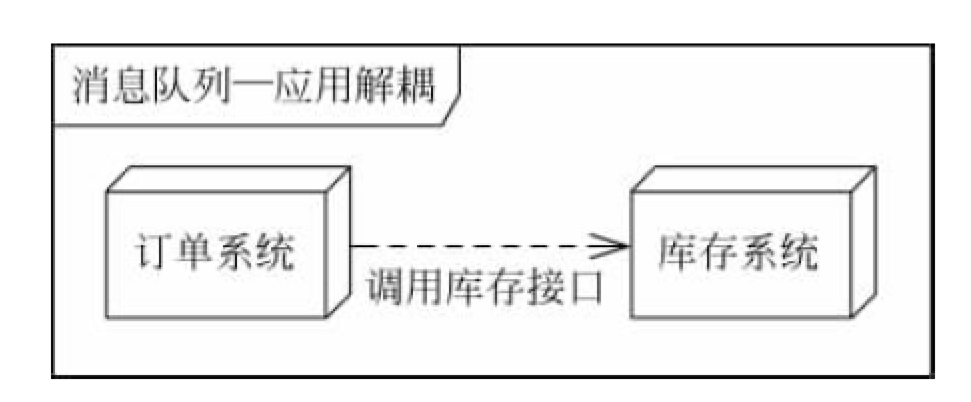


图 25-9　购物网站订单与库存系统

购物网站引入 MQ 消息队列，用户在订单系统上下单成功后，订单系统完成持久化处理，将下订单消息写入 MQ 队列，返回用户订单下单成功。

库存系统从 MQ 队列订阅用户下单的消息，采用拉/推的方式，获取用户下单信息，库

存系统根据用户下单信息，进行减库存操作。

该服务架构的好处是用户下单时库存系统即使不能正常使用，也不影响用户正常下单，因为用户下单后，订单系统将下单消息写入 MQ 队列后就不再关心其他的后续操作，从而实现订单系统与库存系统的应用解耦，如图 25-10 所示。

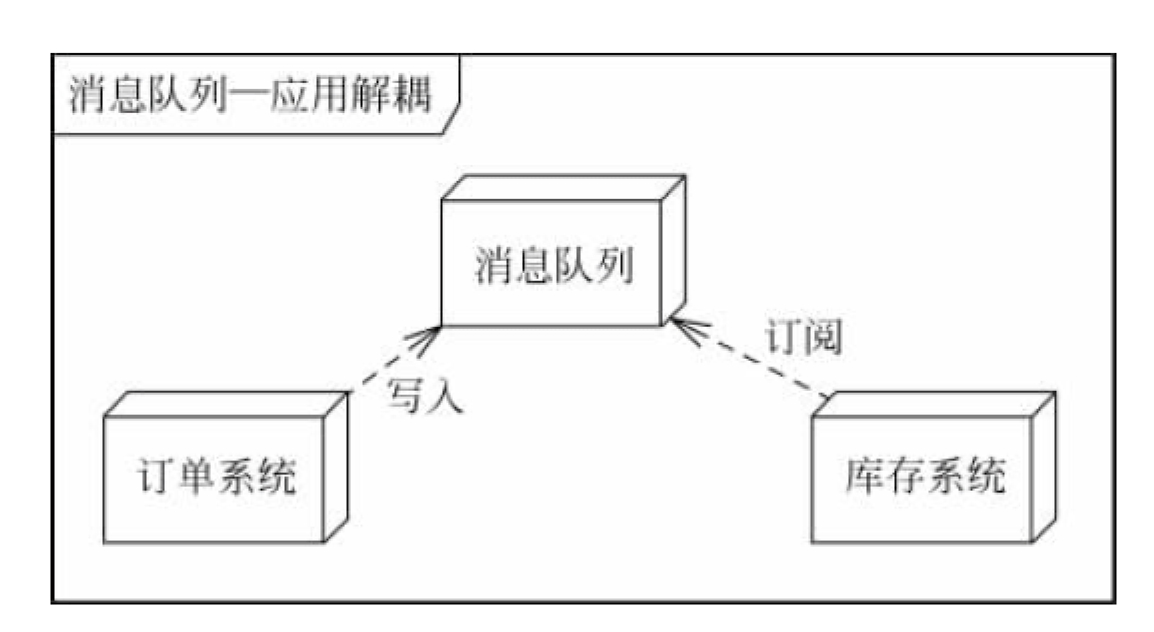


图 25-10 网站订单与库存系统引入 MQ 队列

3. MQ 流量削锋场景

流量削锋也是消息队列中的常用场景，在大型互联网电子商务平台，经常会进行商品秒杀或团购活动，在秒杀或团购活动中，一般会因为流量过大，导致流量暴增，应用挂掉。

架构师为解决这个问题，一般会在应用前端加入 MQ 消息队列，其优点可以控制活动的人数，可以缓解短时间内高流量压垮应用，如图 25-11 所示。

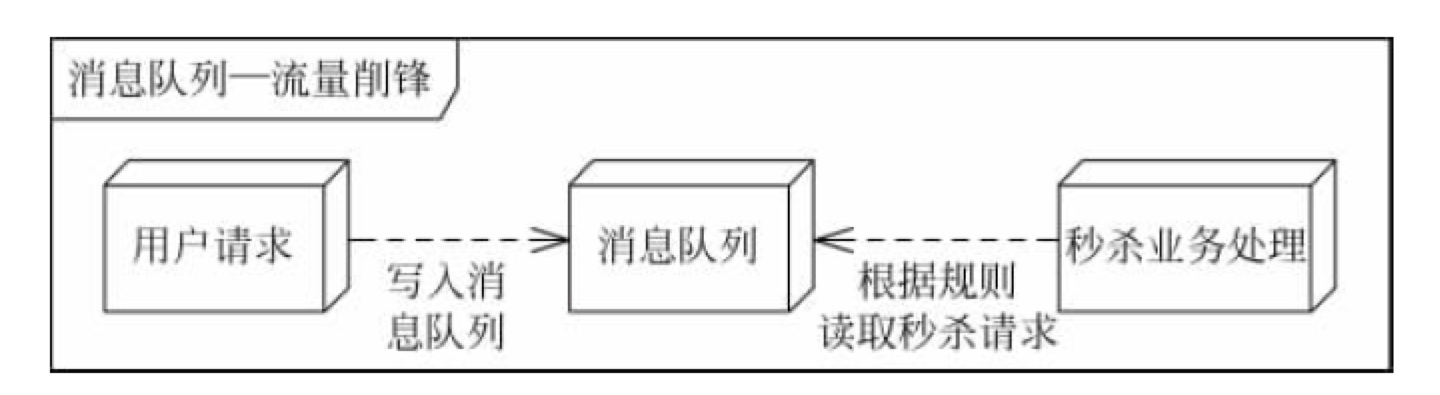


图 25-11 网站秒杀活动服务架构

当用户请求 Web 网站，服务器接收秒杀活动的请求后，首先将用户秒杀的消息写入 MQ 消息队列，假如 MQ 消息队列长度超过设置的秒杀活动最大数量，则直接告诉用户秒杀已经结束、抛弃用户请求或跳转到错误页面，后端秒杀业务根据之前写入 MQ 消息队列中的请求信息，再做后续秒杀成功的用户商品处理。

25.6.3 安装配置 RabbitMQ

安装 RabbitMQ 服务，并启动该服务，RabbitMQ 默认监听端口为 TCP 5672，同时添加 Openstack 用户、添加 RabbitMQ 管理插件，配置步骤如下：

```
#启动 RabbitMQ 服务
systemctl enable rabbitmq-server.service
```

```
systemctl start rabbitmq-server.service
#添加 Openstack 用户和密码,并设置访问权限
rabbitmqctl add_user openstack openstack
rabbitmqctl set_permissions openstack ".*" ".*" ".*"
#查看 RabbitMQ 支持插件,并且启用插件
rabbitmq-plugins list
rabbitmq-plugins enable rabbitmq_management
#重启 RabbitMQ 服务,查看监听端口是否为 15672
systemctl restart rabbitmq-server.service
lsof -i :15672
```

访问 RabbitMQ Web 管理界面，浏览器访问地址为 http://192.168.1.120:15672，访问效果如图 25-12 所示。



图 25-12　RabbitMQ 登录界面

通过默认用户名/密码 guest 登录，添加 Openstack 用户到组，如图 25-13 所示。

创建完毕，使用 Openstack 用户和密码登录，如图 25-14 所示。

25.6.4　RabbitMQ 消息测试

RabbitMQ 消息测试如下，以上消息服务器部署完毕后，可以进行简单消息的发布和订阅。RabbitMQ 完整的消息通信包括：

- 发布者(producer)是发布消息的应用程序；
- 队列(queue)用于消息存储的缓冲；
- 消费者(consumer)是接收消息的应用程序。

RabbitMQ 消息模型核心理念是发布者(producer)不会直接发送任何消息给队列，甚至不知道消息是否已经被投递到队列，而只需要把消息发送给一个交换器(exchange)，交换器非常简单，它可以一边从发布者方接收消息，一边把消息推入队列。交换器必须知道如何处理它接收到的消息，是应该推送到指定的队列还是多个队列，或是直接忽略消息。RabbitMQ Web 操作界面如图 25-15 所示。

```
systemctl start rabbitmq-server.service
#添加 Openstack 用户和密码,并设置访问权限
rabbitmqctl add_user openstack openstack
rabbitmqctl set_permissions openstack ".*" ".*" ".*"
#查看 RabbitMQ 支持插件,并且启用插件
rabbitmq-plugins list
rabbitmq-plugins enable rabbitmq_management
#重启 RabbitMQ 服务,查看监听端口是否为 15672
systemctl restart rabbitmq-server.service
lsof -i :15672
```

访问 RabbitMQ Web 管理界面，浏览器访问地址为 http://192.168.1.120:15672，访问效果如图 25-12 所示。



图 25-12　RabbitMQ 登录界面

通过默认用户名/密码 guest 登录，添加 Openstack 用户到组，如图 25-13 所示。

创建完毕，使用 Openstack 用户和密码登录，如图 25-14 所示。

25.6.4　RabbitMQ 消息测试

RabbitMQ 消息测试如下，以上消息服务器部署完毕后，可以进行简单消息的发布和订阅。RabbitMQ 完整的消息通信包括：

- 发布者(producer)是发布消息的应用程序；
- 队列(queue)用于消息存储的缓冲；
- 消费者(consumer)是接收消息的应用程序。

RabbitMQ 消息模型核心理念是发布者(producer)不会直接发送任何消息给队列，甚至不知道消息是否已经被投递到队列，而只需要把消息发送给一个交换器(exchange)，交换器非常简单，它可以一边从发布者方接收消息，一边把消息推入队列。交换器必须知道如何处理它接收到的消息，是应该推送到指定的队列还是多个队列，或是直接忽略消息。RabbitMQ Web 操作界面如图 25-15 所示。

```
systemctl start rabbitmq-server.service
#添加 Openstack 用户和密码,并设置访问权限
rabbitmqctl add_user openstack openstack
rabbitmqctl set_permissions openstack ".*" ".*" ".*"
#查看 RabbitMQ 支持插件,并且启用插件
rabbitmq-plugins list
rabbitmq-plugins enable rabbitmq_management
#重启 RabbitMQ 服务,查看监听端口是否为 15672
systemctl restart rabbitmq-server.service
lsof -i :15672
```

访问 RabbitMQ Web 管理界面，浏览器访问地址为 http://192.168.1.120:15672，访问效果如图 25-12 所示。



图 25-12 RabbitMQ 登录界面

通过默认用户名/密码 guest 登录，添加 Openstack 用户到组，如图 25-13 所示。

创建完毕，使用 Openstack 用户和密码登录，如图 25-14 所示。

25.6.4 RabbitMQ 消息测试

RabbitMQ 消息测试如下，以上消息服务器部署完毕后，可以进行简单消息的发布和订阅。RabbitMQ 完整的消息通信包括：

- 发布者(producer)是发布消息的应用程序；
- 队列(queue)用于消息存储的缓冲；
- 消费者(consumer)是接收消息的应用程序。

RabbitMQ 消息模型核心理念是发布者(producer)不会直接发送任何消息给队列，甚至不知道消息是否已经被投递到队列，而只需要把消息发送给一个交换器(exchange)，交换器非常简单，它可以一边从发布者方接收消息，一边把消息推入队列。交换器必须知道如何处理它接收到的消息，是应该推送到指定的队列还是多个队列，或是直接忽略消息。RabbitMQ Web 操作界面如图 25-15 所示。

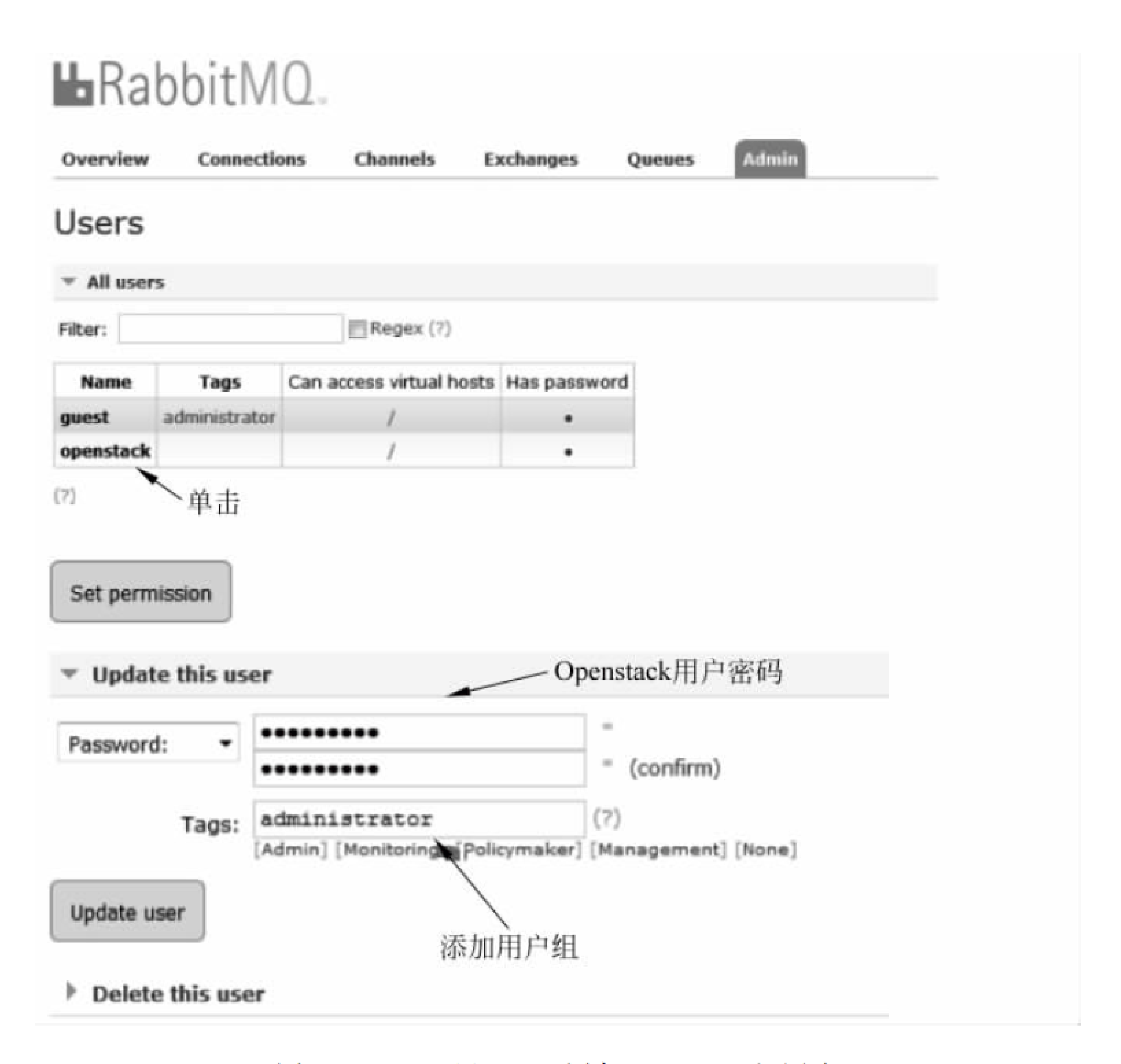


图 25-13 RabbitMQ 添加 Openstack 用户

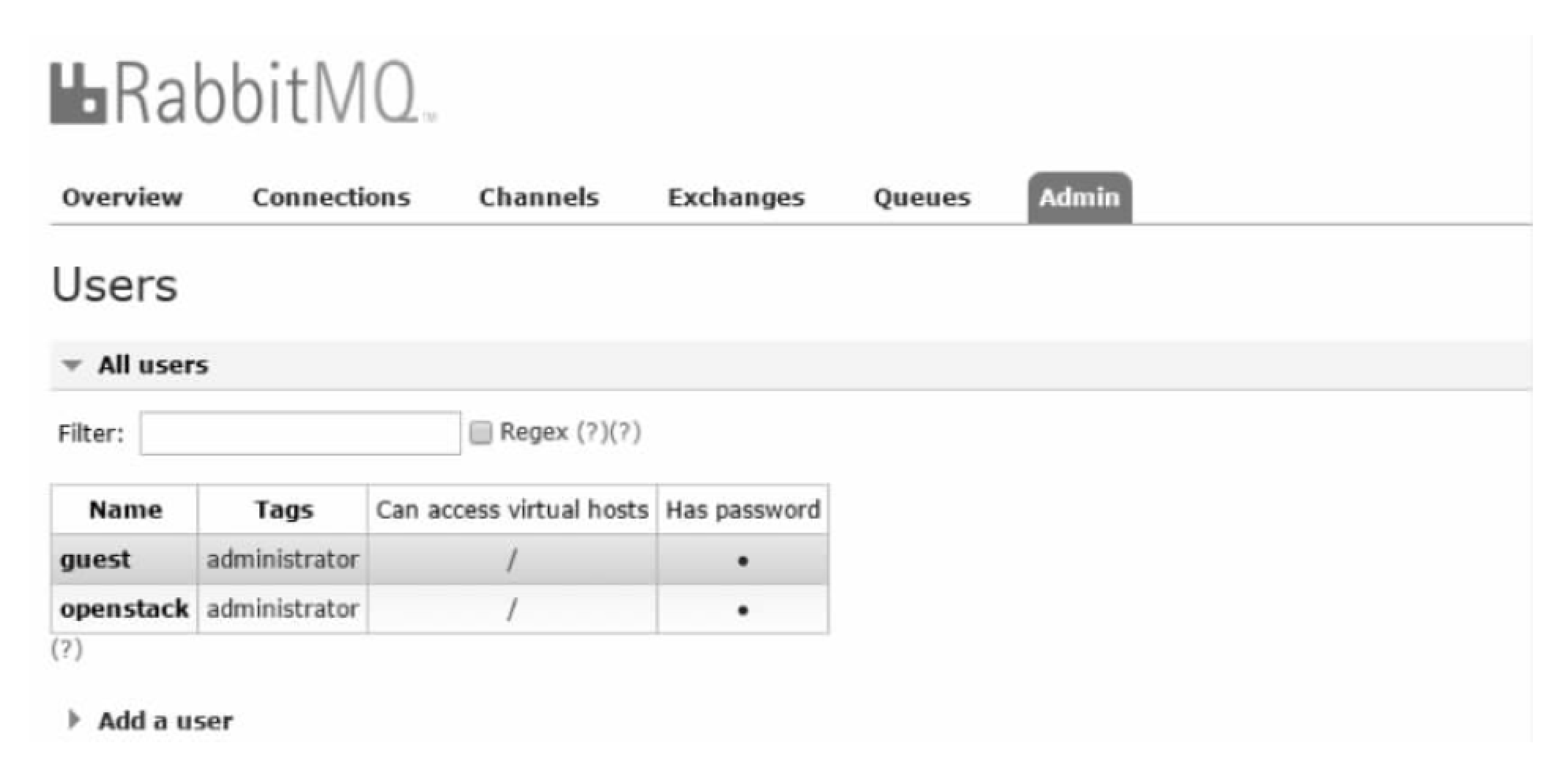
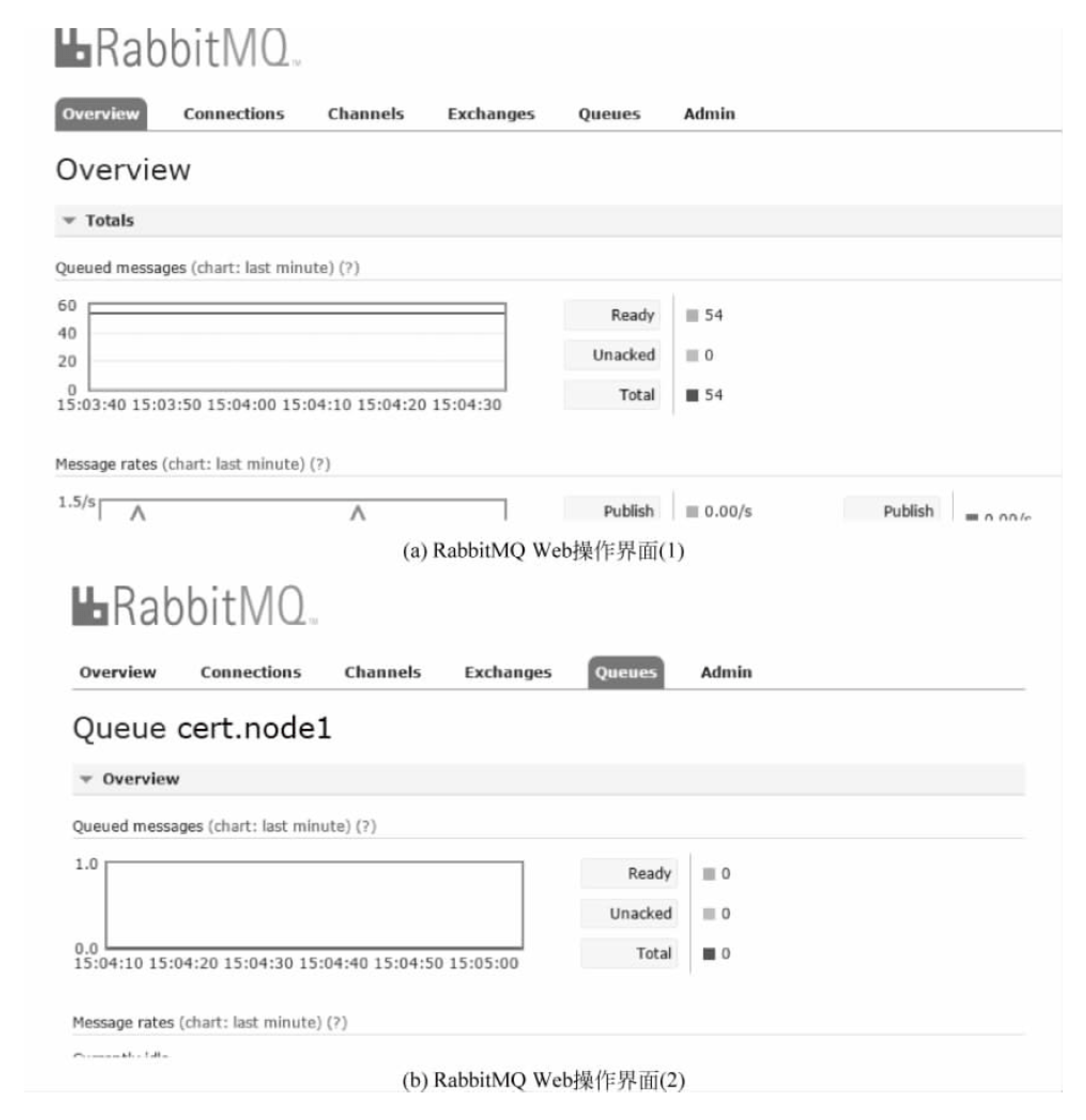


图 25-14 Openstack 用户登录 RabbitMQ



(a) RabbitMQ Web操作界面(1)

(b) RabbitMQ Web操作界面(2)

图 25-15 RabbitMQ Web 操作界面

25.7 配置 Keystone 验证服务

Keystone 身份认证组件是 Openstack 项目中默认的身份认证管理系统，主要用于提供认证服务，所有的服务都需要 Keystone 认证、根据用户的等级分配相应的权限，在 Keystone 中主要涉及以下几个概念：user、tenant、role、token 等，概念详解如下：

- user，使用服务的用户，可以是人、服务或是系统，只要是使用了 Openstack 服务的对象都可以称为用户；
- tenant，租户，可以理解为一个人、项目或组织拥有的资源的合集，在一个租户中可以

拥有很多个用户，这些用户可以根据权限的划分使用租户中的资源；

- role，角色，用于分配操作的权限，角色可以被指定给用户，使得该用户获得角色对应的操作权限；
- token，相当于令牌，是一串比特值或字符串，用来作为访问资源的记号，token 中含有可访问资源的范围和有效时间。

Keystone 身份认证服务架构如图 25-16 所示。

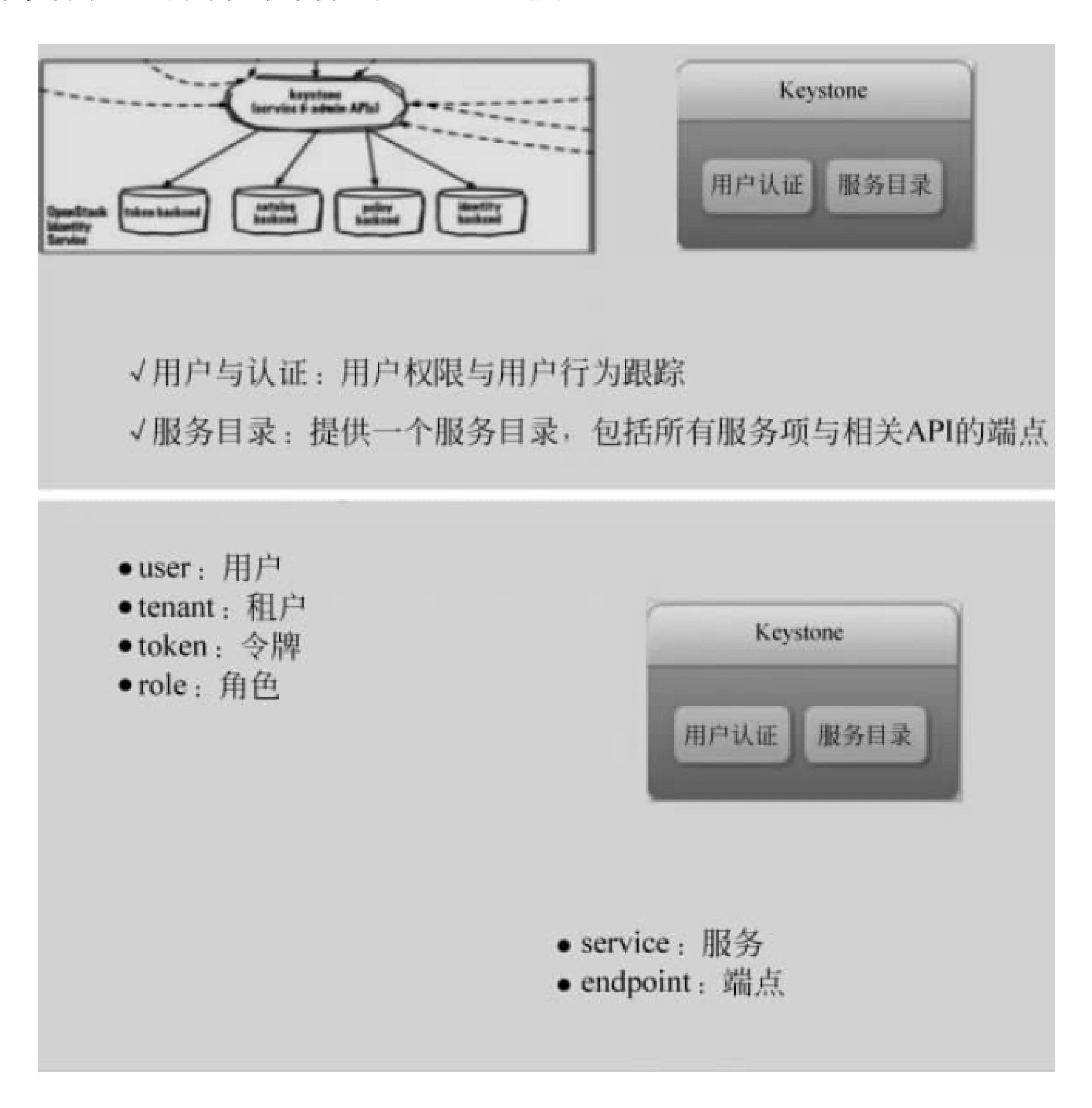


图 25-16 Keystone 认证服务

以下为 node 1 节点上配置 Keystone 服务步骤：

```
#基于 openssl 生成随机数并设置为 admin_token 值;
ID='openssl rand - hex 10';echo $ID
#创建 Keystone 配置文件,并连接数据库、memcached 缓存;
cat >/etc/keystone/keystone.conf << EOF
[DEFAULT]
admin_token = $ID
verbose = true
[assignment]
[auth]
[cache]
[catalog]
[cors]
```

```
[cors.subdomain]
[credential]
[database]
connection = mysql://keystone:keystone@192.168.1.120/keystone
[domain_config]
[endpoint_filter]
[endpoint_policy]
[eventlet_server]
[eventlet_server_ssl]
[federation]
[fernet_tokens]
[identity]
[identity_mapping]
[kvs]
[ldap]
[matchmaker_redis]
[matchmaker_ring]
[memcache]
servers = 192.168.1.120:11211
[oauth1]
[os_inherit]
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
[oslo_middleware]
[oslo_policy]
[paste_deploy]
[policy]
[resource]
[revoke]
driver = sql
[role]
[saml]
[signing]
[ssl]
[token]
provider = uuid
driver = memcache
[tokenless_auth]
[trust]
EOF
#创建数据库表,初始化 Keystone 库并查看日志
su -s /bin/sh -c "keystone-manage db_sync" keystone
tail -n 10 /var/log/keystone/keystone.log
```

配置并启动 memcached 和 Apache 服务，代码如下：

```
#修改 memcached 监听为 0.0.0.0,重启 memcached 服务
sed -i 's/OPTIONS=.*/OPTIONS=\"0.0.0.0\"/g' /etc/sysconfig/memcached
systemctl enable memcached
systemctl start memcached
#配置 Keystone 认证服务接口
cat >/etc/httpd/conf.d/wsgi-keystone.conf << EOF
Listen 5000
Listen 35357
<VirtualHost *:5000>
WSGIDaemonProcess keystone-public processes = 5 threads = 1 user = keystone group = keystone
display-name = %{GROUP}
WSGIProcessGroup keystone-public
WSGIScriptAlias / /usr/bin/keystone-wsgi-public
WSGIApplicationGroup %{GLOBAL}
WSGIPassAuthorization On
<IfVersion >= 2.4>
ErrorLogFormat "%{cu}t %M"
</IfVersion>
ErrorLog /var/log/httpd/keystone-error.log
CustomLog /var/log/httpd/keystone-access.log combined
<Directory /usr/bin>
<IfVersion >= 2.4>
Require all granted
</IfVersion>
<IfVersion < 2.4>
Order allow,deny
Allow from all
</IfVersion>
</Directory>
</VirtualHost>
<VirtualHost *:35357>
WSGIDaemonProcess keystone-admin processes = 5 threads = 1 user = keystone group = keystone
display-name = %{GROUP}
WSGIProcessGroup keystone-admin
WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
WSGIApplicationGroup %{GLOBAL}
WSGIPassAuthorization On
<IfVersion >= 2.4>
ErrorLogFormat "%{cu}t %M"
</IfVersion>
ErrorLog /var/log/httpd/keystone-error.log
CustomLog /var/log/httpd/keystone-access.log combined
<Directory /usr/bin>
<IfVersion >= 2.4>
Require all granted
</IfVersion>
<IfVersion < 2.4>
```

```
Order allow,deny
Allow from all
</IfVersion>
</Directory>
</VirtualHost>
EOF
systemctl enable httpd
systemctl restart httpd
netstat -lntup|grep httpd
```

创建 Keystone 认证用户，临时设置 admin_token 用户的环境变量，用来创建用户，代码如下：

```
export OS_TOKEN= $ID
export OS_URL=http://192.168.1.120:35357/v3
export OS_IDENTITY_API_VERSION=3
#创建 admin 项目
openstack project create --domain default --description "Admin Project" admin
#创建 admin 用户，密码为 admin
openstack user create --domain default --password-prompt admin
```

创建一个普通用户 demo，密码为 demo，代码如下：

```
#创建 admin 角色
openstack role create admin
#将 admin 用户加入到 admin 项目，赋予 admin 的角色
openstack role add --project admin --user admin admin
#同上，创建一个 demo 项目，demo 用户和 demo 角色
openstack project create --domain default --description "Demo Project" demo
openstack user create --domain default --password=demo demo
openstack role create user
openstack role add --project demo --user demo user
#创建 service 项目，用来管理其他服务
openstack project create --domain default --description "Service Project" service
```

检查 Openstack 用户、项目是否正常，代码如下，结果如图 25-17 所示。

```
openstack user list
openstack project list
```

注册 Keystone 访问服务，分别注册为公共的、内部的、管理的类型，即创建 endpoint 服务访问入口，Nova、Swift 和 Glance 一样，每个 Openstack 服务都拥有一个指定的端口和专属的 URL，称该 URL 为访问入口（endpoints），代码如下：

```
openstack service create --name keystone --description "OpenStack Identity" identity
openstack endpoint create --region RegionOne identity public http://192.168.1.120:5000/v2.0
openstack endpoint create --region RegionOne identity internal http://192.168.1.120:5000/v2.0
openstack endpoint create --region RegionOne identity admin http://192.168.1.120:35357/v2.0
```



```
[root@node1 ~]# openstack user list
openstack project list
+----------------------------------+---------+
| ID                               | Name    |
+----------------------------------+---------+
| 4db7d18e9f80470b9a06713374b565e9 | demo    |
| 8c7cd98fc2834fc7b0698405ea6b099b | neutron |
| cbda414d2ea0444a807f98c8f2a0990e | admin   |
| d8a827ad7097482dab6017f25902c41c | glance  |
| e71eb2fd9d7f4e3e850a8b9607662658 | nova    |
+----------------------------------+---------+
[root@node1 ~]# openstack project list
+----------------------------------+---------+
| ID                               | Name    |
+----------------------------------+---------+
| 27b67feaa0d14c0d8b310923404ed493 | admin   |
| 590c7de4a8e54810b2c10a190e68d192 | service |
| e435d0a25c2247f480daef375cfac04d | demo    |
+----------------------------------+---------+
```

图 25-17　Openstack 用户及项目

查看访问入口服务，命令为 openstack endpoint list，结果如图 25-18 所示。

```
[root@node1 ~]# openstack endpoint list #查看
+----------------------------------+-----------+--------------+-------------
-----------------------+
| ID                               | Region    | Service Name | Service Type
                       |
+----------------------------------+-----------+--------------+-------------
-----------------------+
| 10703c763f474bd1bbec86450bd48979 | RegionOne | keystone     | identity
:5000/v2.0
| 22f01b832edc441a8a3eb1ce6c4ce435 | RegionOne | nova         | compute
:8774/v2/%(tenant_id)s |
| 339b9ec5b6454841ad322189f788de97 | RegionOne | glance       | image
:9292
| 33aad9a889174d869db1da405c36d9f0 | RegionOne | glance       | image
:9292
| 67aa2437cbb44814a8d2fffe75eb0897 | RegionOne | nova         | compute
:8774/v2/%(tenant_id)s |
| 6e69e7bc50c24751b5e2c582f62b9d33 | RegionOne | neutron      | network
:9696
| 7f08b9bd301b4565a2e3b1157001a8aa | RegionOne | keystone     | identity
```

图 25-18　Openstack endpoint 接口

验证默认项目、认证用户，获取 token 值，只有获取到项目 ID、用户 ID 等信息才能说明 Keystone 配置成功，代码如下，结果如图 25-19 所示。

```
#unset OS_TOKEN
#unset OS_URL
openstack -- os - auth - url http://192. 168. 1. 120: 35357/v3  -- os - project - domain - id
default -- os - user - domain - id default -- os - project - name admin -- os - username admin -
- os - auth - type password token issue
```

```
[root@node1 ~]#
[root@node1 ~]#
[root@node1 ~]# unset OS_TOKEN
[root@node1 ~]# unset OS_URL
[root@node1 ~]# openstack --os-auth-url http://192.168.1.120:3
default --os-project-name admin --os-username admin --os-auth-
+------------+----------------------------------+
| Field      | Value                            |
+------------+----------------------------------+
| expires    | 2017-07-06T11:42:45.597581Z      |
| id         | b3c4d93d5d3e42968ab9d9727e65f54f |
| project_id | 27b67feaa0d14c0d8b310923404ed493 |
| user_id    | cbda414d2ea0444a807f98c8f2a0990e |
+------------+----------------------------------+
[root@node1 ~]#
```

图 25-19　Openstack token 值验证

创建项目、用户及租户管理配置文件，使用环境变量来获取 token，环境变量创建虚拟机时需要调用，配置代码如下：

```
cat > admin - openrc. sh << EOF
export OS_PROJECT_DOMAIN_ID = default
export OS_USER_DOMAIN_ID = default
export OS_PROJECT_NAME = admin
export OS_TENANT_NAME = admin
export OS_USERNAME = admin
export OS_PASSWORD = admin
export OS_AUTH_URL = http://192.168.1.120:35357/v3
export OS_IDENTITY_API_VERSION = 3
EOF
cat > demo - openrc. sh << EOF
export OS_PROJECT_DOMAIN_ID = default
export OS_USER_DOMAIN_ID = default
export OS_PROJECT_NAME = demo
export OS_TENANT_NAME = demo
export OS_USERNAME = demo
export OS_PASSWORD = demo
export OS_AUTH_URL = http://192.168.1.120:5000/v3
export OS_IDENTITY_API_VERSION = 3
EOF
unset OS_TOKEN
unset OS_URL
```

引用 shell 脚本，验证默认项目、认证用户，获取 token 值，命令如下，访问如图 25-20 所示。

```
source admin - openrc. sh
openstack token issue
```

```
192.168.1.120 × | 192.168.1.120 (1) | 192.168.1.121 | 192.168.1.121 (1)
[root@node1 ~]#
[root@node1 ~]# openstack token issue
+------------+----------------------------------+
| Field      | Value                            |
+------------+----------------------------------+
| expires    | 2017-07-06T11:45:36.998607Z      |
| id         | 7ad8b85881d34de08d5873e42019224a |
| project_id | 27b67feaa0d14c0d8b310923404ed493 |
| user_id    | cbda414d2ea0444a807f98c8f2a0990e |
+------------+----------------------------------+
[root@node1 ~]#
```

图 25-20　Openstack 脚本 token 值验证

25.8　配置 Glance 镜像服务

Glance 是 Openstack 镜像服务组件，提供虚拟机镜像的发现、注册、取得服务等功能，Glance 提供 restful API 可以查询虚拟机镜像 metadata 信息，并且可以直接获取虚拟机镜像，为创建虚拟机提供底层镜像模板支持。

通过 Glance 组件，虚拟机镜像可以被存储到多种存储上，比如简单的硬盘存储、对象存储或分布式文件系统等，Glance 组件内容及工作流程，如图 25-21 所示。



图 25-21　Openstack Glance 组件功能

以下为 Glance 组件服务配置方法，vim 修改 Glance API 接口配置文件/etc/glance/glance-api.conf，设置内容如下：

```
cat >/etc/glance/glance-api.conf << EOF
[DEFAULT]
verbose=True
notification_driver = noop
#Galnce 不需要消息队列
[database]
connection=mysql://glance:glance@192.168.1.120/glance
[glance_store]
default_store=file
#虚拟机镜像存储位置
filesystem_store_datadir=/var/lib/glance/images/
[image_format]
[keystone_authtoken]
auth_uri = http://192.168.1.120:5000
auth_url = http://192.168.1.120:35357
auth_plugin = password
project_domain_id = default
```

```
user_domain_id = default
project_name = service
username = glance
password = glance
[matchmaker_redis]
[matchmaker_ring]
[oslo_concurrency]
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
[oslo_policy]
[paste_deploy]
flavor = keystone
[store_type_location_strategy]
[task]
[taskflow_executor]
EOF
```

修改 Glance 镜像注册配置文件/etc/glance/glance-registry.conf，内容如下：

```
cat >/etc/glance/glance-registry.conf << EOF
[DEFAULT]
verbose = True
notification_driver = noop
[database]
connection = mysql://glance:glance@192.168.1.120/glance
[glance_store]
[keystone_authtoken]
auth_uri = http://192.168.1.120:5000
auth_url = http://192.168.1.120:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = glance
password = glance
[matchmaker_redis]
[matchmaker_ring]
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
[oslo_policy]
[paste_deploy]
flavor = keystone
EOF
```

创建 Glance 数据库表，同步 Glance 数据库，代码如下：

```
su -s /bin/sh -c "glance-manage db_sync" glance
```

创建 Glance 的 Keystone 用户，注册用户信息，代码如下：

```
source admin-openrc.sh
openstack user create --domain default --password=glance glance
openstack role add --project service --user glance admin
#启动 Glance API 及 registry 服务
systemctl enable openstack-glance-api
systemctl enable openstack-glance-registry
systemctl start openstack-glance-api
systemctl start openstack-glance-registry
#查看 Glance API 及 registry 服务进程端口
netstat -lnutp |grep 9191 #registry
netstat -lnutp |grep 9292 #api
```

在 Keystone 认证中心注册 Glance 镜像服务，创建镜像访问入口服务，代码如下：

```
Source admin-openrc.sh
openstack service create --name glance --description "OpenStack Image service" image
openstack endpoint create --region RegionOne image public http://192.168.1.120:9292
openstack endpoint create --region RegionOne image internal http://192.168.1.120:9292
openstack endpoint create --region RegionOne image admin http://192.168.1.120:9292
```

添加 Glance 环境变量，获取镜像列表，代码如下：

```
echo "export OS_IMAGE_API_VERSION=2" | tee -a admin-openrc.sh demo-openrc.sh
glance image-list
```

Wget 远程下载 cirros 或者 CentOS 7 镜像，上传到 Glance 存储仓库，代码如下：

```
wget -q http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
glance image-create --name "cirros" --file cirros-0.3.4-x86_64-disk.img --disk-
format qcow2 --container-format bare --visibility public --progress
wget http://cloud.CentOS.org/CentOS/7/images/CentOS-7-x86_64-GenericCloud.qcow2
glance image-create --name "CentOS-7-x86_64" --file CentOS-7-x86_64-GenericCloud.
qcow2 --disk-format qcow2 --container-format bare --visibility public --progress
```

查看 Glance 镜像仓库列表，代码如下，结果如图 25-22 所示。

```
glance image-list
ll /var/lib/glance/images/
```

```
[root@node1 ~]# netstat -lnutp |grep 9292 #api
tcp        0      0 0.0.0.0:9292            0.0.0.0:*               LISTEN
[root@node1 ~]# glance image-list
ll /var/lib/glance/images/
+--------------------------------------+------------------+
| ID                                   | Name             |
+--------------------------------------+------------------+
| a8e32bcb-0b05-4cf9-ba7a-f4d0caee291a | CentOS-7-x86_64  |
| a6f3ef4f-059c-462c-af35-99e507659921 | centos7          |
| 71da6aae-9609-4f79-ba55-c4af7e2428e1 | cirros           |
+--------------------------------------+------------------+
[root@node1 ~]# ll /var/lib/glance/images/
total 2061812
-rw-r----- 1 glance glance   13287936 Jul  7  2017 71da6aae-9609-4f79-ba55
-rw-r----- 1 glance glance  713031680 Jul  5 15:53 a6f3ef4f-059c-462c-af35
-rw-r----- 1 glance glance 1384972288 Jul  5 02:19 a8e32bcb-0b05-4cf9-ba7a
[root@node1 ~]#
```

图 25-22　Openstack Glance 镜像列表

25.9 Nova 控制节点配置

Nova 是 Openstack 框架中最重要的调度和组织控制器，支持 Openstack 私有云实例(instances)生命周期的所有管理操作，可以把 Nova 看成是一个负责管理计算资源、网络、认证可扩展性的平台。

Nova 自身并没有提供任何虚拟化能力，它使用 libvirt API 来与被支持的 Hypervisors 虚拟化引擎进行交互，实现虚拟机的创建、销毁、IP 配置、管理、重启等操作，Nova 各个模块功能如图 25-23 所示。

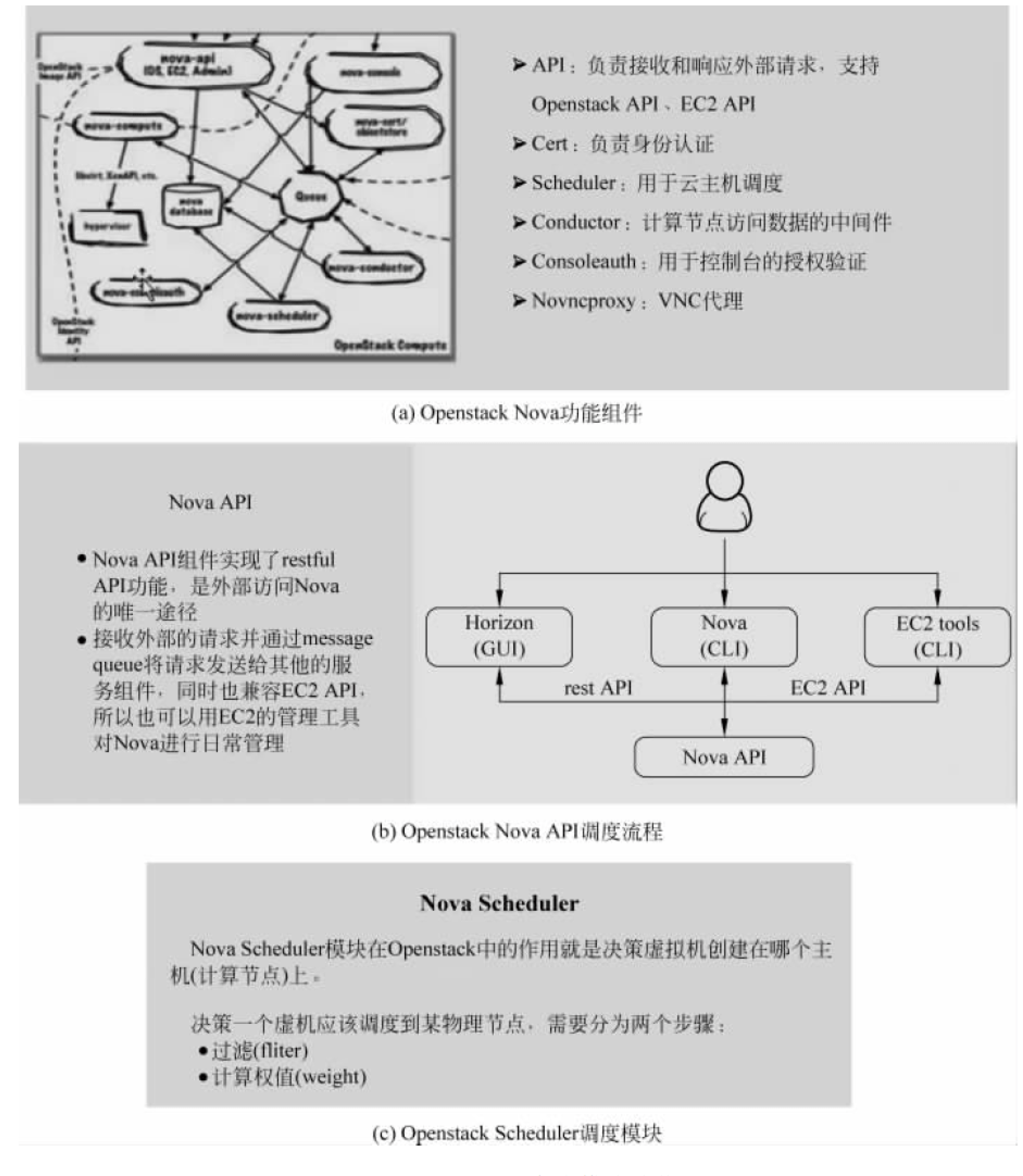
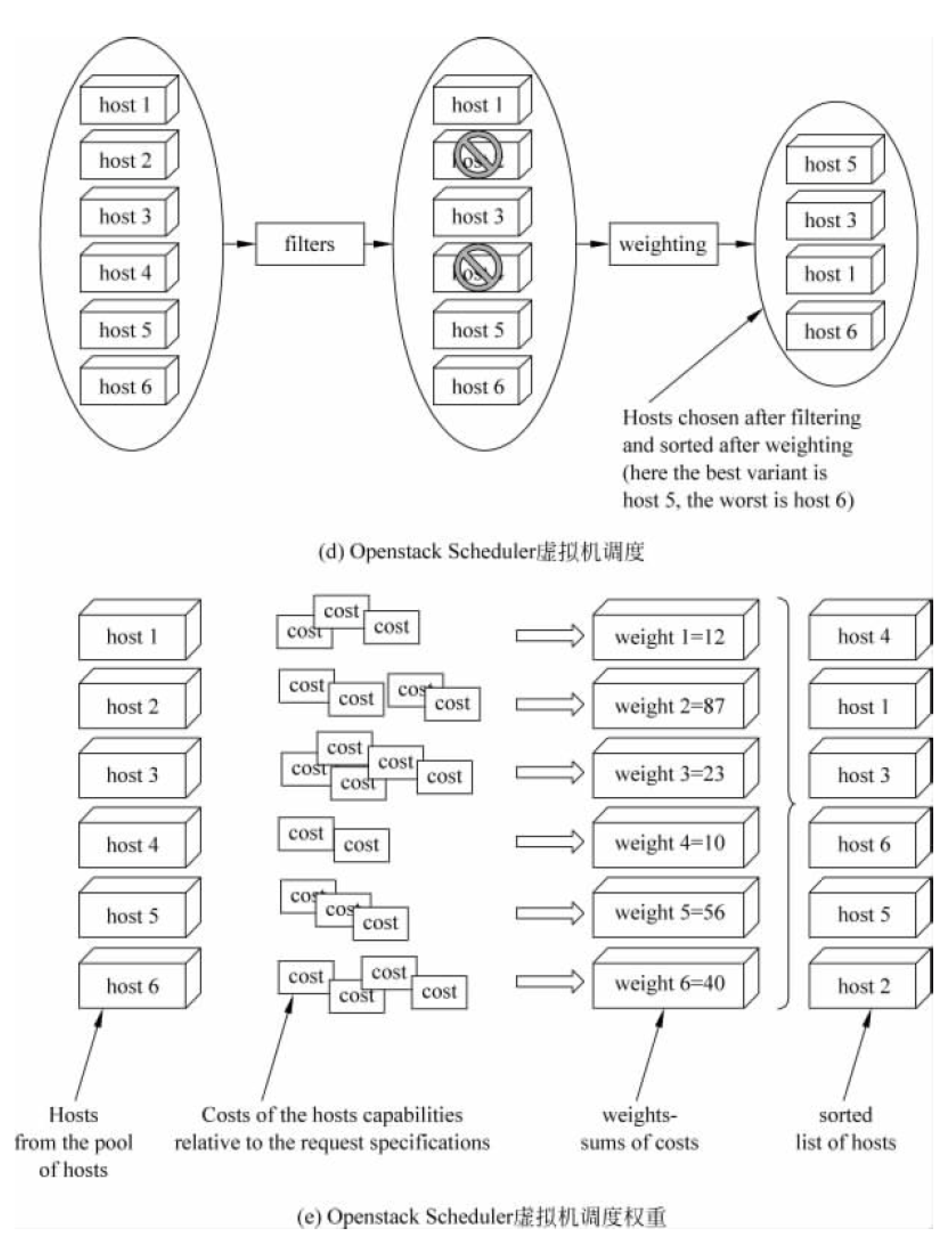


图 25-23　Nova 各个模块功能



(d) Openstack Scheduler虚拟机调度

(e) Openstack Scheduler虚拟机调度权重

图 25-23 （续）

配置 Openstack Nova 组件服务，配置代码如下：

```
cat >/etc/nova/nova.conf << EOF
[DEFAULT]
my_ip = 192.168.1.120
enabled_apis = osapi_compute,metadata
auth_strategy = keystone
network_api_class = nova.network.neutronv2.api.API
linuxnet_interface_driver = nova.network.linux_net.NeutronLinuxBridgeInterfaceDriver
security_group_api = neutron
```

```
firewall_driver = nova.virt.firewall.NoopFirewallDriver
debug = true
#vif_plugging_is_fatal = False
#vif_plugging_timeout = 0
verbose = true
rpc_backend = rabbit
allow_resize_to_same_host = True
scheduler_default_filters = RetryFilter, AvailabilityZoneFilter, RamFilter, ComputeFilter,
ComputeCapabilitiesFilter,ImagePropertiesFilter,ServerGroupAntiAffinityFilter,ServerGroup
AffinityFilter
[api_database]
[barbican]
[cells]
[cinder]
[conductor]
[cors]
[cors.subdomain]
[database]
connection = mysql://nova:nova@192.168.1.120/nova
[ephemeral_storage_encryption]
[glance]
host = \$my_ip
[guestfs]
[hyperv]
[image_file_url]
[ironic]
[keymgr]
[keystone_authtoken]
auth_uri = http://192.168.1.120:5000
auth_url = http://192.168.1.120:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
password = nova
[libvirt]
virt_type = qemu
#virt_type = kvm
[matchmaker_redis]
[matchmaker_ring]
[metrics]
[neutron]
url = http://192.168.1.120:9696
auth_url = http://192.168.1.120:35357
auth_plugin = password
project_domain_id = default
```

```
user_domain_id = default
region_name = RegionOne
project_name = service
username = neutron
password = neutron
service_metadata_proxy = True
metadata_proxy_shared_secret = neutron
lock_path = /var/lib/nova/tmp
[osapi_v21]
[oslo_concurrency]
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
rabbit_host = 192.168.1.120
rabbit_port = 5672
rabbit_userid = openstack
rabbit_password = openstack
[oslo_middleware]
[rdp]
[serial_console]
[spice]
[ssl]
[trusted_computing]
[upgrade_levels]
[vmware]
[vnc]
novncproxy_base_url = http://192.168.1.120:6080/vnc_auto.html
vncserver_listen = \$my_ip
vncserver_proxyclient_address = \$my_ip
keymap = en-us
[workarounds]
[xenserver]
[zookeeper]
EOF
```

同步 Nova 数据库并创建 Nova 用户及启动服务，代码如下：

```
su -s /bin/sh -c "nova-manage db sync" nova
#创建 Nova Keystone 用户
openstack user create --domain default --password = nova nova
openstack role add --project service --user nova admin
#启动 Nova 相关服务
systemctl enable openstack-nova-api.service openstack-nova-cert.service openstack-nova-consoleauth.service openstack-nova-scheduler.service openstack-nova-conductor.service openstack-nova-novncproxy.service
systemctl restart openstack-nova-api.service openstack-nova-cert.service openstack-nova-consoleauth.service openstack-nova-scheduler.service openstack-nova-conductor.
```

```
service openstack - nova - novncproxy.service
#在 Keystone 注册中心中注册 Nova 访问入口服务
source admin - openrc.sh
openstack service create -- name nova -- description "OpenStack Compute" compute
openstack endpoint create -- region RegionOne compute public http://192.168.1.120:8774/v2/%
\(tenant_id\)s
openstack endpoint create -- region RegionOne compute internal http://192.168.1.120:8774/
v2/%\(tenant_id\)s
openstack endpoint create -- region RegionOne compute admin http://192.168.1.120:8774/v2/%\
(tenant_id\)s
```

至此，Openstack 控制节点 Nova 服务配置完毕，通过命令 openstack host list 查看 Openstack 主机列表，如图 25-24 所示。

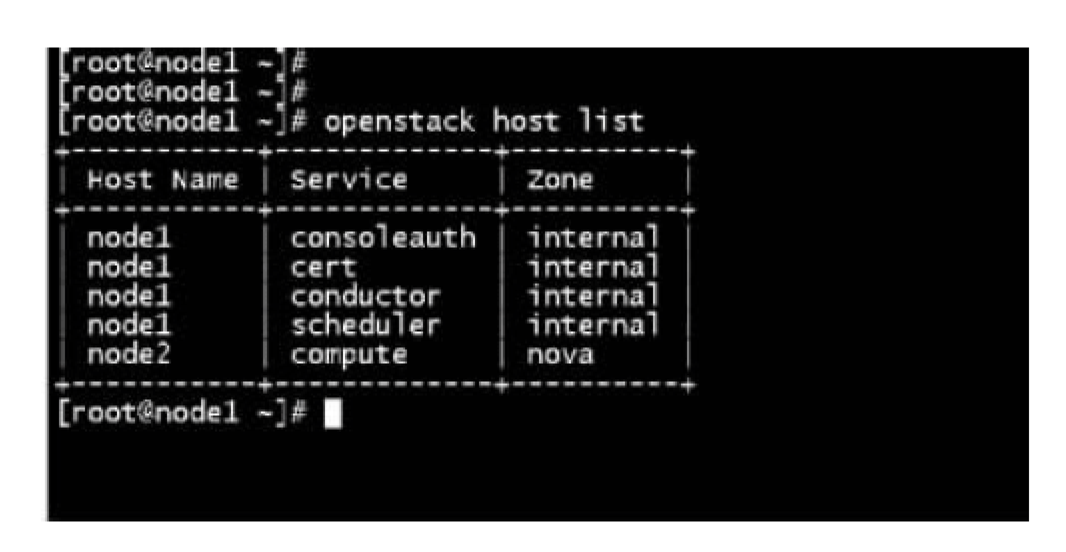


图 25-24　Openstack 主机节点列表

25.10　Nova 计算节点配置

控制节点 Nova 配置完毕后，需要配置计算节点 Nova 服务，计算节点 Nova compute 主要是接收 MQ 管理 VM 的消息，通过 libvirt API 对虚拟机进行管理，如图 25-25 所示。

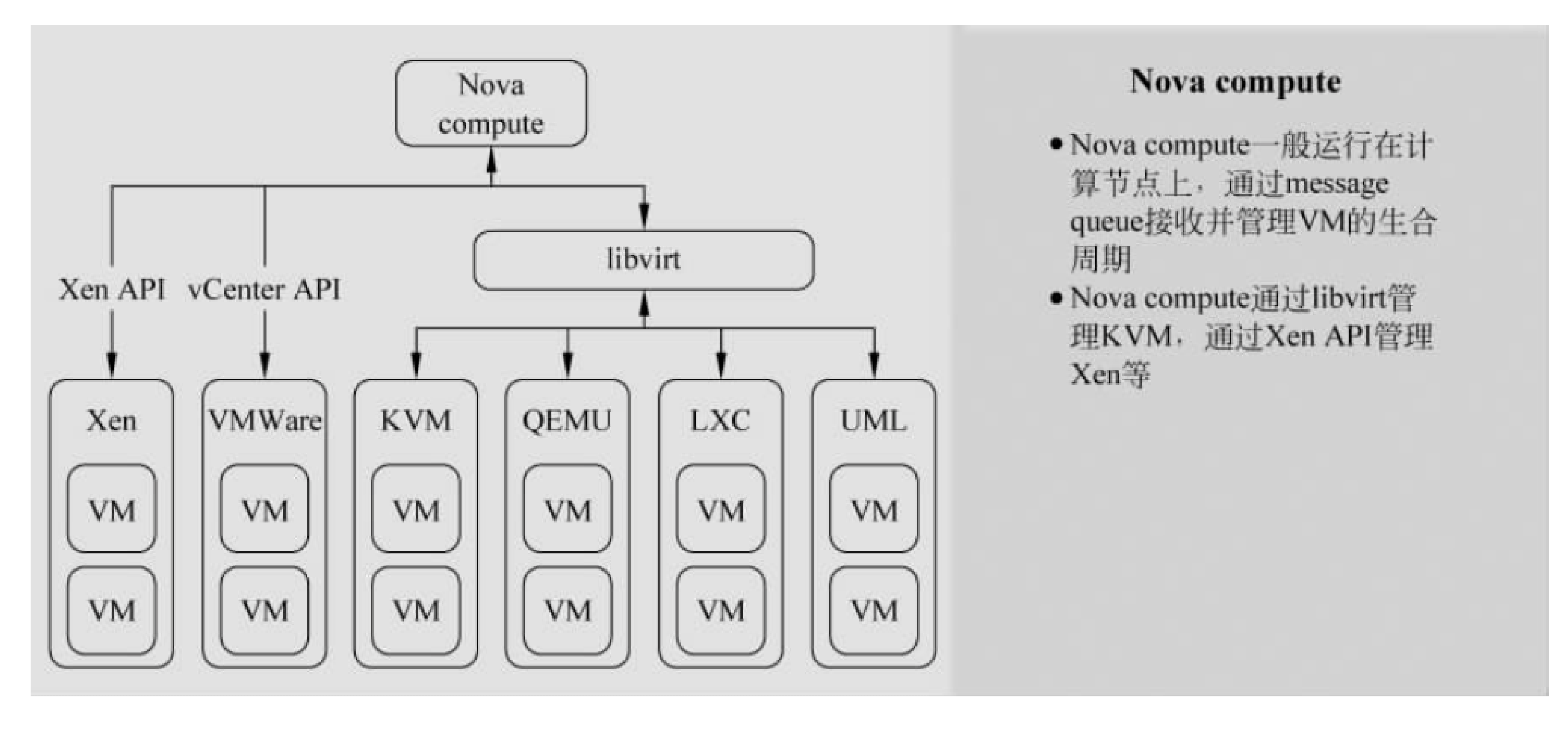


图 25-25　Openstack 计算节点 Nova 组件

node 2 Nova 计算节点上安装 Nova 相关软件包,代码如下:

```
#安装 epel 及 Openstack liberty 版本 YUM 源
yum install -y epel-release
yum install -y CentOS-release-openstack-liberty
yum install -y python-openstackclient
#安装 Openstack Nova 计算节点服务
yum install -y openstack-nova-compute sysfsutils
#安装 Openstack Neutron 网络管理服务
yum install -y openstack-neutron openstack-neutron-linuxbridge ebtables ipset
#安装 Openstack Cinder 块设备存储模块
yum install -y openstack-cinder python-cinderclient targetcli python-oslo-policy
#升级 KVM Qemu 管理工具
yum install -y CentOS-release-qemu-ev.noarch
yum install qemu-kvm qemu-img -y
```

计算节点 vim 修改 nova.conf 主配置文件,代码如下:

```
cat > /etc/nova/nova.conf << EOF
[DEFAULT]
my_ip = 192.168.1.121
enabled_apis = osapi_compute,metadata
auth_strategy = keystone
network_api_class = nova.network.neutronv2.api.API
linuxnet_interface_driver = nova.network.linux_net.NeutronLinuxBridgeInterfaceDriver
security_group_api = neutron
firewall_driver = nova.virt.firewall.NoopFirewallDriver
debug = true
#vif_plugging_is_fatal = False
#vif_plugging_timeout = 0
verbose = true
rpc_backend = rabbit
allow_resize_to_same_host = True
scheduler_default_filters = RetryFilter, AvailabilityZoneFilter, RamFilter, ComputeFilter,
ComputeCapabilitiesFilter,ImagePropertiesFilter,ServerGroupAntiAffinityFilter,ServerGroup
AffinityFilter
[api_database]
[barbican]
[cells]
[cinder]
[conductor]
[cors]
[cors.subdomain]
[database]
connection = mysql://nova:nova@192.168.1.120/nova
[ephemeral_storage_encryption]
[glance]
```

```
host = 192.168.1.120
[guestfs]
[hyperv]
[image_file_url]
[ironic]
[keymgr]
[keystone_authtoken]
auth_uri = http://192.168.1.120:5000
auth_url = http://192.168.1.120:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
password = nova
[libvirt]
virt_type = qemu
#virt_type = kvm
[matchmaker_redis]
[matchmaker_ring]
[metrics]
[neutron]
url = http://192.168.1.120:9696
auth_url = http://192.168.1.120:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
region_name = RegionOne
project_name = service
username = neutron
password = neutron
service_metadata_proxy = True
metadata_proxy_shared_secret = neutron
lock_path = /var/lib/nova/tmp
[osapi_v21]
[oslo_concurrency]
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
rabbit_host = 192.168.1.120
rabbit_port = 5672
rabbit_userid = openstack
rabbit_password = openstack
[oslo_middleware]
[rdp]
[serial_console]
[spice]
```

```
[ssl]
[trusted_computing]
[upgrade_levels]
[vmware]
[vnc]
novncproxy_base_url = http://192.168.1.120:6080/vnc_auto.html
vncserver_listen =  0.0.0.0
vncserver_proxyclient_address = \ $my_ip
keymap = en - us
[workarounds]
[xenserver]
[zookeeper]
EOF
```

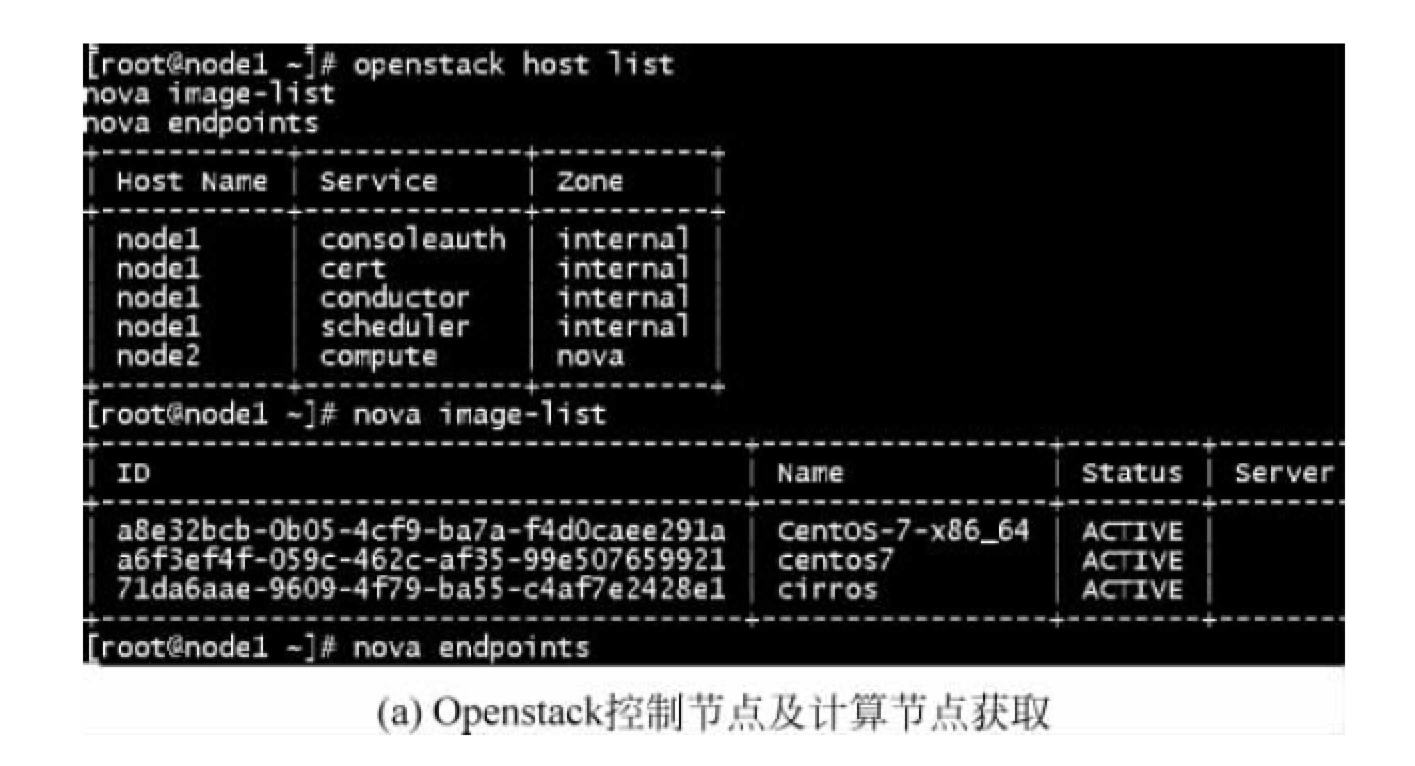
启动计算节点 Nova libvirtd 及 Nova 服务，命令如下：

```
systemctl enable libvirtd openstack - nova - compute
systemctl restart libvirtd openstack - nova - compute
```

25.11　Openstack 节点测试

Nova 控制节点与计算节点配置完毕，以下为在 node 1 计算节点进行测试，测试命令如下，测试结果如图 25-26 所示。

```
#获取 Openstack 主节点及计算节点
openstack host list
#获取 Glance 仓库镜像
glance image - list
#通过 Nova API 接口获取 Glance 镜像
nova image - list
```



```
[root@node1 ~]# openstack host list
nova image-list
nova endpoints
+-----------+-------------+----------+
| Host Name | Service     | Zone     |
+-----------+-------------+----------+
| node1     | consoleauth | internal |
| node1     | cert        | internal |
| node1     | conductor   | internal |
| node1     | scheduler   | internal |
| node2     | compute     | nova     |
+-----------+-------------+----------+
[root@node1 ~]# nova image-list
+--------------------------------------+-----------------+--------+--------+
| ID                                   | Name            | Status | Server |
+--------------------------------------+-----------------+--------+--------+
| a8e32bcb-0b05-4cf9-ba7a-f4d0caee291a | CentOS-7-x86_64 | ACTIVE |        |
| a6f3ef4f-059c-462c-af35-99e507659921 | centos7         | ACTIVE |        |
| 71da6aae-9609-4f79-ba55-c4af7e2428e1 | cirros          | ACTIVE |        |
+--------------------------------------+-----------------+--------+--------+
[root@node1 ~]# nova endpoints
```

(a) Openstack控制节点及计算节点获取

图 25-26　Openstack 节点测试



(b) Openstack endpoint访问接口(1)

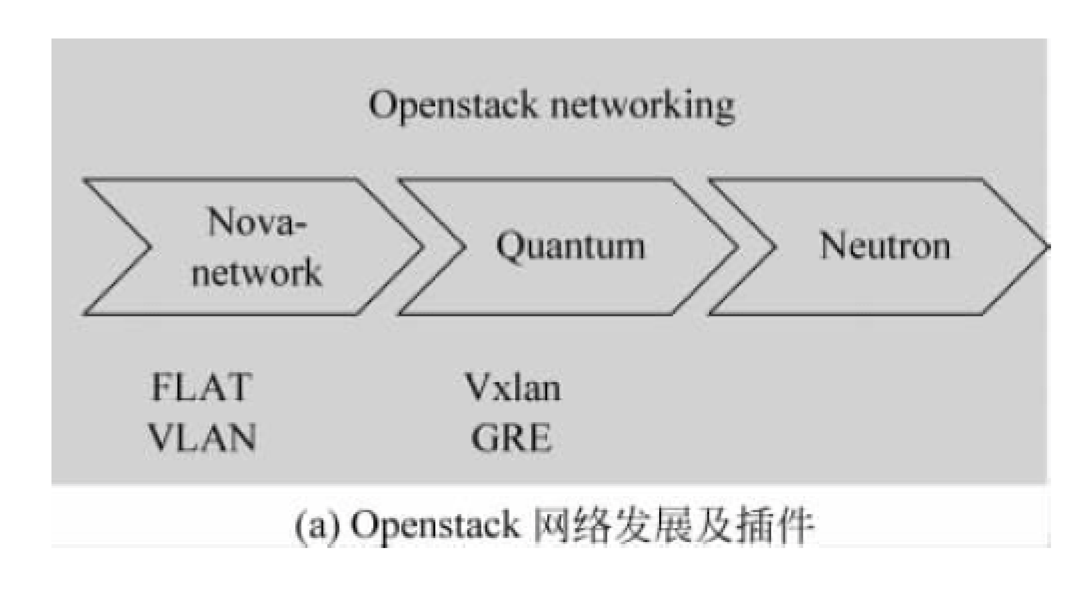
(c) Openstack endpoint访问接口(2)

图 25-26 （续）

25.12 Neutron 控制节点配置

Openstack Neutron 组件主要提供云计算的网络虚拟化功能，为 Openstack 其他服务提供网络连接服务，为用户提供接口，可以定义 network、subnet、router、DHCP、DNS、负载均衡、L3 服务、GRE、VLAN，插件架构支持许多主流的网络厂家和技术。

Openstack 网络发展经历过 3 次大的变革和发展，从最早的 Nova-network 到 Quantum，再到最新的 Neutron，支持的网络插件越来越多，Openstack 网络发展及组件功能如图 25-27 所示。



(a) Openstack 网络发展及插件

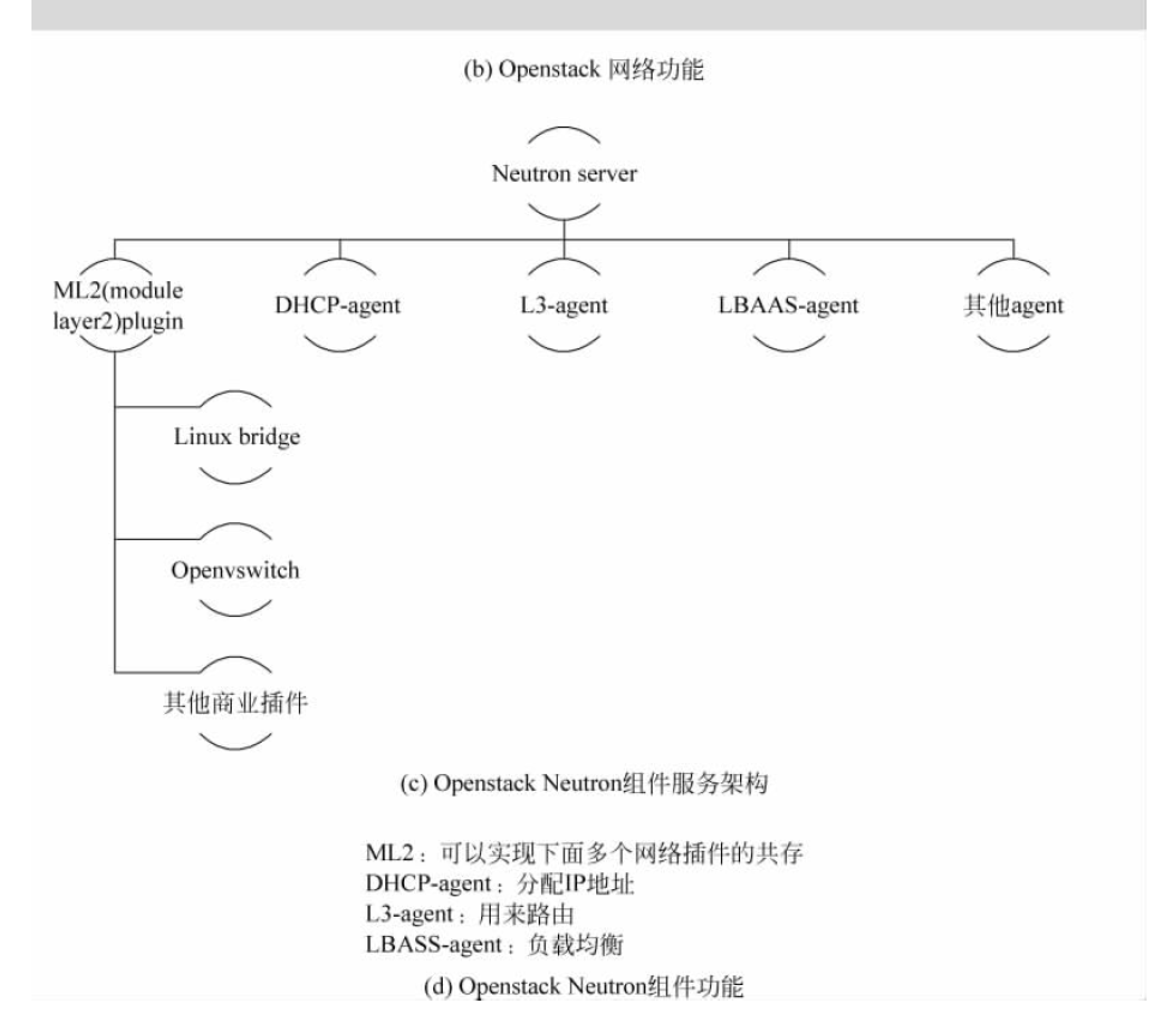
图 25-27 Openstack 网络发展及组件功能

公共网络：向租户提供访问或者API调用
管理网络：云中物理机之间的通信
存储网络：云中存储的网络，如ISCSI或GlusterFS使用
服务网络：虚拟机内部使用的网络

Openstack networking

- 网络：在实际的物理环境下，一般使用交换机或集线器把多个计算机连接起来形成网络。在Neutron的世界里，网络也是将多个不同的云主机连接起来
- 子网：在实际的物理环境下，在一个网络中，可以将网络划分成多个逻辑子网，在Neutron的世界里，子网也是隶属于网络下的
- 端口：在实际的物理环境下，每个子网或每个网络，都有很多的端口，比如交换机端口来供计算机连接，在Neutron的世界里，端口也是隶属于子网下，云主机的网卡会对应到一个端口上
- 路由器：在实际的网络环境，不同网络或不同逻辑子网之间如果需要进行通信，需要通过路由器进行路由，在Neutron的世界里，路由也是这个作用，用来连接不同的网络或子网

(b) Openstack 网络功能



(c) Openstack Neutron组件服务架构

ML2：可以实现下面多个网络插件的共存
DHCP-agent：分配IP地址
L3-agent：用来路由
LBASS-agent：负载均衡

(d) Openstack Neutron组件功能

图 25-27 （续）

Neutron 控制节点 node 1 配置，修改/etc/neutron/neutron.conf 主配置文件，执行如下命令：

```
cat >/etc/neutron/neutron.conf << EOF
[DEFAULT]
state_path = /var/lib/neutron
core_plugin = ml2
service_plugins = router
auth_strategy = keystone
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://192.168.1.120:8774/v2
rpc_backend = rabbit
[matchmaker_redis]
[matchmaker_ring]
[quotas]
[agent]
[keystone_authtoken]
auth_uri = http://192.168.1.120:5000
auth_url = http://192.168.1.120:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = neutron
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%
[database]
connection = mysql://neutron:neutron@192.168.1.120:3306/neutron
[nova]
auth_url = http://192.168.1.120:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
region_name = RegionOne
project_name = service
username = nova
password = nova
[oslo_concurrency]
lock_path = $state_path/lock
[oslo_policy]
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
rabbit_host = 192.168.1.120
```

```
rabbit_port = 5672
rabbit_userid = openstack
rabbit_password = openstack
[qos]
EOF
```

创建配置文件 ml2_conf.ini、linuxbridge_agent.ini、dhcp_agent.ini、metadata_agent.ini，配置方法如下：

```
cat >/etc/neutron/plugins/ml2/ml2_conf.ini << EOF
[ml2]
type_drivers = flat,vlan,gre,vxlan,geneve
tenant_network_types = vlan,gre,vxlan,geneve
mechanism_drivers = openvswitch,linuxbridge
extension_drivers = port_security
[ml2_type_flat]
flat_networks = physnet1
[ml2_type_vlan]
[ml2_type_gre]
[ml2_type_vxlan]
[ml2_type_geneve]
[securitygroup]
enable_ipset = True
EOF
cat >/etc/neutron/plugins/ml2/linuxbridge_agent.ini << EOF
[linux_bridge]
physical_interface_mappings = physnet1:eth0
[vxlan]
enable_vxlan = false
[agent]
prevent_arp_spoofing = True
[securitygroup]
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
enable_security_group = True
EOF
cat >/etc/neutron/dhcp_agent.ini << EOF
[DEFAULT]
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = true
[AGENT]
EOF
cat >/etc/neutron/metadata_agent.ini << EOF
[DEFAULT]
auth_uri = http://192.168.1.120:5000
auth_url = http://192.168.1.120:35357
auth_region = RegionOne
```

```
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = neutron
nova_metadata_ip = 192.168.1.120
metadata_proxy_shared_secret = neutron
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%
[AGENT]
EOF
```

创建 Keystone 的用户 Neutron，更新数据库信息及注册至 Keystone 中，代码如下：

```
#创建连接并创建 Keystone 的用户
ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
openstack user create --domain default --password=neutron neutron
openstack role add --project service --user neutron admin
#更新 Neutron 数据库
su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-
file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config
-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
#在 Keystone 注册中心注册网络访问入口服务
source admin-openrc.sh
openstack service create --name neutron --description "OpenStack Networking" network
openstack endpoint create --region RegionOne network public http://192.168.1.120:9696
openstack endpoint create --region RegionOne network internal http://192.168.1.120:9696
openstack endpoint create --region RegionOne network admin http://192.168.1.120:9696
```

因为 Neutron 和 Nova 需要关联，对 Neutron 进行调整，需要重启 openstack-nova-api 接口服务，Nova 相关服务重启命令如下：

```
systemctl restart openstack-nova-api.service openstack-nova-cert.service openstack-
nova-consoleauth.service openstack-nova-scheduler.service openstack-nova-conductor.
service openstack-nova-novncproxy.service
systemctl enable neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp
-agent.service neutron-metadata-agent.service
systemctl restart neutron-server.service neutron-linuxbridge-agent.service neutron-dhcp
-agent.service neutron-metadata-agent.service
mkdir -p /lock;chmod 777 -R /lock
```

Neutron 节点配置测试，代码如下，结果如图 25-28 所示。

```
#查看 Openstack 所有访问入口服务
openstack endpoint list
```

```
[root@node1 ~]# openstack endpoint list
+----------------------------------+-----------+--------------+--------------+
-------------------------+
| ID                               | Region    | Service Name | Service Type |
                         |
+----------------------------------+-----------+--------------+--------------+
-------------------------+
| 10703c763f474bd1bbec86450bd48979 | RegionOne | keystone     | identity     |
:5000/v2.0               |
| 22f01b832edc441a8a3eb1ce6c4ce435 | RegionOne | nova         | compute      |
:8774/v2/%(tenant_id)s   |
| 339b9ec5b6454841ad322189f788de97 | RegionOne | glance       | image        |
:9292                    |
| 33aad9a889174d869db1da405c36d9f0 | RegionOne | glance       | image        |
:9292                    |
| 67aa2437cbb44814a8d2fffe75eb0897 | RegionOne | nova         | compute      |
:8774/v2/%(tenant_id)s   |
| 6e69e7bc50c24751b5e2c582f62b9d33 | RegionOne | neutron      | network      |
:9696                    |
| 7f08b9bd301b4565a2e3b1157001a8aa | RegionOne | keystone     | identity     |
```

图 25-28　Openstack 访问入口服务

25.13　Neutron 计算节点配置

Neutron 除了控制节点 node 1 配置之外，在每台计算节点同样需要进行配置，修改 node 2 主配置文件/etc/neutron/neutron.conf 内容如下：

```
cat >/etc/neutron/neutron.conf << EOF
[DEFAULT]
state_path = /var/lib/neutron
core_plugin = ml2
service_plugins = router
auth_strategy = keystone
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://192.168.1.120:8774/v2
rpc_backend = rabbit
[matchmaker_redis]
[matchmaker_ring]
[quotas]
[agent]
[keystone_authtoken]
auth_uri = http://192.168.1.120:5000
auth_url = http://192.168.1.120:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = neutron
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%
```

```
[database]
connection = mysql://neutron:neutron@192.168.1.120:3306/neutron
[nova]
auth_url = http://192.168.1.120:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
region_name = RegionOne
project_name = service
username = nova
password = nova
[oslo_concurrency]
lock_path = $state_path/lock
[oslo_policy]
[oslo_messaging_amqp]
[oslo_messaging_qpid]
[oslo_messaging_rabbit]
rabbit_host = 192.168.1.120
rabbit_port = 5672
rabbit_userid = openstack
rabbit_password = openstack
[qos]
EOF
```

修改 Neutron 相关配置文件 linuxbridge_agent.ini、ml2_conf.ini，配置方法如下：

```
cat >/etc/neutron/plugins/ml2/linuxbridge_agent.ini << EOF
[linux_bridge]
physical_interface_mappings = physnet1:eth0
[vxlan]
enable_vxlan = false
[agent]
prevent_arp_spoofing = True
[securitygroup]
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
enable_security_group = True
EOF
cat >/etc/neutron/plugins/ml2/ml2_conf.ini << EOF
[ml2]
type_drivers = flat,vlan,gre,vxlan,geneve
tenant_network_types = vlan,gre,vxlan,geneve
mechanism_drivers = openvswitch,linuxbridge
extension_drivers = port_security
[ml2_type_flat]
flat_networks = physnet1
[ml2_type_vlan]
```

```
[ml2_type_gre]
[ml2_type_vxlan]
[ml2_type_geneve]
[securitygroup]
enable_ipset = True
EOF
mkdir -p /lock;chmod 777 -R /lock
ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
systemctl enable neutron-linuxbridge-agent.service
systemctl restart neutron-linuxbridge-agent.service
```

25.14　控制节点创建网桥

Neutron 控制节点和计算节点配置完毕，在 node 1 控制节点上检查控制节点及计算节点信息，代码如下，结果如图 25-29 所示。

```
neutron agent-list
```

```
[root@node1 ~]# neutron agent-list
+--------------------------------------+--------------------+-------+-------
+
| id                                   | agent_type         | host  | alive
+--------------------------------------+--------------------+-------+-------
+
| 191c5ff0-4720-4804-bf88-60e29db4832c | Linux bridge agent | node1 | :-)
| 6ba3694d-c590-4fe3-a75f-122c5407c2b0 | Metadata agent     | node1 | :-)
| 97c99b0b-2dba-48c4-b67e-5465577f228f | Linux bridge agent | node2 | :-)
| c74452f5-7b8e-4b7b-961d-2657f91ed8ef | DHCP agent         | node1 | :-)
+--------------------------------------+--------------------+-------+-------
+
```

图 25-29　Openstack 网络节点 agent 信息

在 node 1 控制主节点，创建共享网络，并创建虚拟机分配网段及网关地址，代码如下，结果如图 25-30 所示。

```
source admin-openrc.sh
neutron net-create flat --shared --provider:physical_network physnet1 --provider:
network_type flat
neutron subnet-create flat 192.168.1.0/24 --name flat-subnet --allocation-pool start=
192.168.1.140,end=192.168.1.200 --dns-nameserver 192.168.1.1 --gateway 192.168.1.1
neutron net-list
neutron subnet-list
```

创建 Openstack 密钥对主要是用于免密码登录，默认创建虚拟机，登录虚拟机需要用户名和密码，创建密钥对之后在控制节点登录新创建的虚拟机可以直接以密钥的方式登录，无须密码，因为创建虚拟机时，会将公钥写入到虚拟机系统中，以下为生成密钥对及创建安全组的方法：

```
[root@node1 ~]# neutron net-list
neutron subnet-list
+--------------------------------------+------+-----------------------------------------
| id                                   | name | subnets
+--------------------------------------+------+-----------------------------------------
| 2d642a8c-d0c6-444e-828c-6a988539ddad | flat | a22d1bde-2ff1-4073-9703-6f7eca626bb8 19
+--------------------------------------+------+-----------------------------------------
[root@node1 ~]# neutron subnet-list
+--------------------------------------+-------------+----------------+-----------------
--+
| id                                   | name        | cidr           | allocation_pool
  |
+--------------------------------------+-------------+----------------+-----------------
--+
| a22d1bde-2ff1-4073-9703-6f7eca626bb8 | flat-subnet | 192.168.1.0/24 | {"start": "192.
} |
+--------------------------------------+-------------+----------------+-----------------
--+
```

图 25-30 Openstack 创建 Neutron 虚拟机网络

```
#创建 ssh keypair 密钥对
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
nova keypair-add --pub-key /root/.ssh/id_rsa.pub mykey
nova keypair-list
#创建安全组,允许 icmp ping、22、80 端口
nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
nova secgroup-add-rule default tcp 80 80 0.0.0.0/0
```

查看 Openstack 支持的虚拟机类型、Nova 镜像列表及 Neutron 网络，命令如下，结果如图 25-31 所示。

```
#查看 Openstack 支持的虚拟机类型
nova flavor-list
#查看 Nova 镜像列表
nova image-list
#查看 Neutron 网络
neutron net-list
```

```
| c13fce6e-1210-4395-b8a9-a4b0e415c7a3 | m1.small | 512 | 10
  |
+--------------------------------------+----------+-----+------
--+
[root@node1 ~]# #查看镜像
[root@node1 ~]# nova image-list
+--------------------------------------+---------------+--------+--
| ID                                   | Name          | Status | S
+--------------------------------------+---------------+--------+--
| a8e32bcb-0b05-4cf9-ba7a-f4d0caee291a | CentOS-7-x86_64 | ACTIVE |
| a6f3ef4f-059c-462c-af35-99e507659921 | centos7       | ACTIVE |
| 71da6aae-9609-4f79-ba55-c4af7e2428e1 | cirros        | ACTIVE |
+--------------------------------------+---------------+--------+--
[root@node1 ~]# #查看网络
[root@node1 ~]# neutron net-list
+--------------------------------------+------+-------------------
| id                                   | name | subnets
+--------------------------------------+------+-------------------
| 2d642a8c-d0c6-444e-828c-6a988539ddad | flat | a22d1bde-2ff1-4073-97
+--------------------------------------+------+-------------------
```

图 25-31 Openstack 虚拟机类型、镜像及网络信息

通过 Nova 指令创建虚拟机，指定网络 ID 为 2d64a8c-d0c6-444e-828c-6a988539ddad，指定镜像名称为 cirros，启动命令如下，结果如图 25-32 所示。

```
nova boot --flavor m1.tiny --image cirros --nic net-id=6277d20f-d033-42b8-96bc-6565ff07e8a3 --security-group default --key-name mykey hello-instance
```

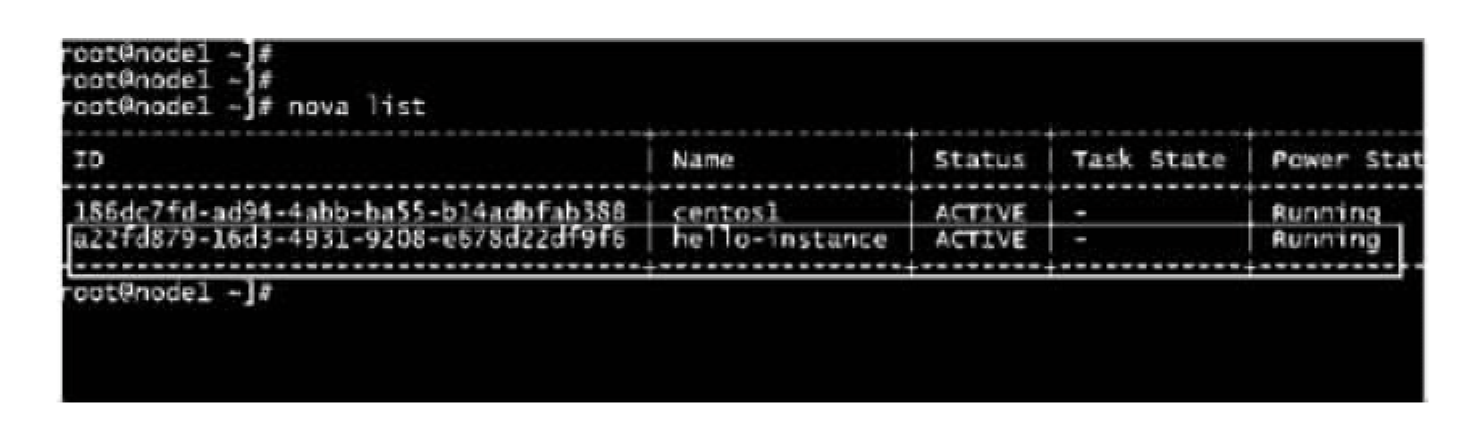


图 25-32　Openstack 虚拟机实例

创建虚拟机的时候，Openstack 在 Neutron 组网内是采用 dhcp-agent 自动分配 IP，可以在创建虚拟机的时候，指定固定的 IP 地址。

25.15　控制节点配置 dashboard

Openstack 各个组件配置完毕，为了能更加方便、快捷地对虚拟机及 Openstack 私有云进行管理，引入 dashboard Web 界面可以在 Web 平台中去管理 Openstack 相关组件，Openstack Web 部署方法如下：

```
yum install openstack-dashboard -y
cat > /etc/openstack-dashboard/local_settings << EOF
import os
from django.utils.translation import ugettext_lazy as _
from openstack_dashboard import exceptions
from openstack_dashboard.settings import HORIZON_CONFIG
DEBUG = False
TEMPLATE_DEBUG = DEBUG
# WEBROOT is the location relative to Webserver root
# should end with a slash.
WEBROOT = '/dashboard/'
# LOGIN_URL = WEBROOT + 'auth/login/'
# LOGOUT_URL = WEBROOT + 'auth/logout/'
# LOGIN_REDIRECT_URL can be used as an alternative for
# HORIZON_CONFIG.user_home, if user_home is not set.
# Do not set it to '/home/', as this will cause circular redirect loop
#LOGIN_REDIRECT_URL = WEBROOT
# Required for Django 1.5.
# If horizon is running in production (DEBUG is False), set this
# with the list of host/domain names that the application can serve.
# For more information see:
# https://docs.djangoproject.com/en/dev/ref/settings/#allowed-hosts
ALLOWED_HOSTS = ['*']
# Set SSL proxy settings:
```

```
# For Django 1.4+ pass this header from the proxy after terminating the SSL,
# and don't forget to strip it from the client's request.
# For more information see:
# https://docs.djangoproject.com/en/1.4/ref/settings/#secure-proxy-ssl-header
#SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTOCOL', 'https')
# https://docs.djangoproject.com/en/1.5/ref/settings/#secure-proxy-ssl-header
#SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
# If Horizon is being served through SSL, then uncomment the following two
# settings to better secure the cookies from security exploits
#CSRF_COOKIE_SECURE = True
#SESSION_COOKIE_SECURE = True
# Overrides for OpenStack API versions. Use this setting to force the
# OpenStack dashboard to use a specific API version for a given service API.
# Versions specified here should be integers or floats, not strings.
# NOTE: The version should be formatted as it appears in the URL for the
# service API. For example, The identity service APIs have inconsistent
# use of the decimal point, so valid options would be 2.0 or 3.
#OPENSTACK_API_VERSIONS = {
#     "data-processing": 1.1,
#     "identity": 3,
#     "volume": 2,
#}

# Set this to True if running on multi-domain model. When this is enabled, it
# will require user to enter the Domain name in addition to username for login.
#OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = False
# Overrides the default domain used when running on single-domain model
# with Keystone V3. All entities will be created in the default domain.
#OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = 'Default'
# Set Console type:
# valid options are "AUTO"(default), "VNC", "SPICE", "RDP", "SERIAL" or None
# Set to None explicitly if you want to deactivate the console.
#CONSOLE_TYPE = "AUTO"
# Show backdrop element outside the modal, do not close the modal
# after clicking on backdrop.
#HORIZON_CONFIG["modal_backdrop"] = "static"
# Specify a regular expression to validate user passwords.
#HORIZON_CONFIG["password_validator"] = {
#     "regex": '.*',
#     "help_text": _("Your password does not meet the requirements."),
#}
# Disable simplified floating IP address management for deployments with
# multiple floating IP pools or complex network requirements.
#HORIZON_CONFIG["simple_ip_management"] = False
# Turn off browser autocompletion for forms including the login form and
# the database creation workflow if so desired.
#HORIZON_CONFIG["password_autocomplete"] = "off"
```

```
# Setting this to True will disable the reveal button for password fields,
# including on the login form.
#HORIZON_CONFIG["disable_password_reveal"] = False
LOCAL_PATH = '/tmp'
# Set custom secret key:
# You can either set it to a specific value or you can let horizon generate a
# default secret key that is unique on this machine, e.i. regardless of the
# amount of Python WSGI workers (if used behind Apache + mod_wsgi): However,
# there may be situations where you would want to set this explicitly, e.g.
# when multiple dashboard instances are distributed on different machines
# (usually behind a load-balancer). Either you have to make sure that a session
# gets all requests routed to the same dashboard instance or you set the same
# SECRET_KEY for all of them.
SECRET_KEY='36c739e5c252f9e014d9'
# We recommend you use memcached for development; otherwise after every reload
# of the django development server, you will have to login again. To use
# memcached set CACHES to something like
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '192.168.1.120:11211',
    }
}
#CACHES = {
#    'default': {
#        'BACKEND': 'django.core.cache.backends.locmem.LocMemCache',
#    }
#}
# Send email to the console by default
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
# Or send them to /dev/null
#EMAIL_BACKEND = 'django.core.mail.backends.dummy.EmailBackend'
# Configure these for your outgoing email host
#EMAIL_HOST = 'smtp.my-company.com'
#EMAIL_PORT = 25
#EMAIL_HOST_USER = 'djangomail'
#EMAIL_HOST_PASSWORD = 'top-secret!'
# For multiple regions uncomment this configuration, and add (endpoint, title).
#AVAILABLE_REGIONS = [
#    ('http://cluster1.example.com:5000/v2.0', 'cluster1'),
#    ('http://cluster2.example.com:5000/v2.0', 'cluster2'),
#]
OPENSTACK_HOST = "192.168.1.120"
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v2.0" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"

# Enables keystone web single-sign-on if set to True.
```

```
#WEBSSO_ENABLED = False
# Determines which authentication choice to show as default.
#WEBSSO_INITIAL_CHOICE = "credentials"
# The list of authentication mechanisms
# which include keystone federation protocols.
# Current supported protocol IDs are 'saml2' and 'oidc'
# which represent SAML 2.0, OpenID Connect respectively.
# Do not remove the mandatory credentials mechanism.
#WEBSSO_CHOICES = (
#     ("credentials", _("Keystone Credentials")),
#     ("oidc", _("OpenID Connect")),
#     ("saml2", _("Security Assertion Markup Language")))
# Disable SSL certificate checks (useful for self-signed certificates):
#OPENSTACK_SSL_NO_VERIFY = True
# The CA certificate to use to verify SSL connections
#OPENSTACK_SSL_CACERT = '/path/to/cacert.pem'
# The OPENSTACK_KEYSTONE_BACKEND settings can be used to identify the
# capabilities of the auth backend for Keystone.
# If Keystone has been configured to use LDAP as the auth backend then set
# can_edit_user to False and name to 'ldap'.
#
# TODO(tres): Remove these once Keystone has an API to identify auth backend.
OPENSTACK_KEYSTONE_BACKEND = {
    'name': 'native',
    'can_edit_user': True,
    'can_edit_group': True,
    'can_edit_project': True,
    'can_edit_domain': True,
    'can_edit_role': True,
}

# Setting this to True, will add a new "Retrieve Password" action on instance,
# allowing Admin session password retrieval/decryption.
#OPENSTACK_ENABLE_PASSWORD_RETRIEVE = False

# The Launch Instance user experience has been significantly enhanced.
# You can choose whether to enable the new launch instance experience,
# the legacy experience, or both. The legacy experience will be removed
# in a future release, but is available as a temporary backup setting to ensure
# compatibility with existing deployments. Further development will not be
# done on the legacy experience. Please report any problems with the new
# experience via the Launchpad tracking system.
#
# Toggle LAUNCH_INSTANCE_LEGACY_ENABLED and LAUNCH_INSTANCE_NG_ENABLED to
# determine the experience to enable. Set them both to true to enable
# both.
#LAUNCH_INSTANCE_LEGACY_ENABLED = True
```

```
#LAUNCH_INSTANCE_NG_ENABLED = False

# The Xen Hypervisor has the ability to set the mount point for volumes
# attached to instances (other Hypervisors currently do not). Setting
# can_set_mount_point to True will add the option to set the mount point
# from the UI.
OPENSTACK_HYPERVISOR_FEATURES = {
    'can_set_mount_point': False,
    'can_set_password': False,
    'requires_keypair': False,
}

# The OPENSTACK_CINDER_FEATURES settings can be used to enable optional
# services provided by cinder that is not exposed by its extension API.
OPENSTACK_CINDER_FEATURES = {
    'enable_backup': False,
}

# The OPENSTACK_NEUTRON_NETWORK settings can be used to enable optional
# services provided by neutron. Options currently available are load
# balancer service, security groups, quotas, VPN service.
OPENSTACK_NEUTRON_NETWORK = {
    'enable_router': True,
    'enable_quotas': True,
    'enable_ipv6': True,
    'enable_distributed_router': False,
    'enable_ha_router': False,
    'enable_lb': True,
    'enable_firewall': True,
    'enable_vpn': True,
    'enable_fip_topology_check': True,

    # Neutron can be configured with a default Subnet Pool to be used for IPv4
    # subnet-allocation. Specify the label you wish to display in the Address
    # pool selector on the create subnet step if you want to use this feature.
    'default_ipv4_subnet_pool_label': None,

    # Neutron can be configured with a default Subnet Pool to be used for IPv6
    # subnet-allocation. Specify the label you wish to display in the Address
    # pool selector on the create subnet step if you want to use this feature.
    # You must set this to enable IPv6 Prefix Delegation in a PD-capable
    # environment.
    'default_ipv6_subnet_pool_label': None,

    # The profile_support option is used to detect if an external router can be
    # configured via the dashboard. When using specific plugins the
    # profile_support can be turned on if needed.
```

```
        'profile_support': None,
        #'profile_support': 'cisco',

        # Set which provider network types are supported. Only the network types
        # in this list will be available to choose from when creating a network.
        # Network types include local, flat, vlan, gre, and vxlan.
        'supported_provider_types': ['*'],

        # Set which VNIC types are supported for port binding. Only the VNIC
        # types in this list will be available to choose from when creating a
        # port.
        # VNIC types include 'normal', 'macvtap' and 'direct'.
        # Set to empty list or None to disable VNIC type selection.
        'supported_vnic_types': ['*']
}

# The OPENSTACK_IMAGE_BACKEND settings can be used to customize features
# in the OpenStack Dashboard related to the Image service, such as the list
# of supported image formats.
#OPENSTACK_IMAGE_BACKEND = {
#     'image_formats': [
#          ('', _('Select format')),
#          ('aki', _('AKI - Amazon Kernel Image')),
#          ('ami', _('AMI - Amazon Machine Image')),
#          ('ari', _('ARI - Amazon Ramdisk Image')),
#          ('docker', _('Docker')),
#          ('iso', _('ISO - Optical Disk Image')),
#          ('ova', _('OVA - Open Virtual Appliance')),
#          ('qcow2', _('QCOW2 - QEMU Emulator')),
#          ('raw', _('Raw')),
#          ('vdi', _('VDI - Virtual Disk Image')),
#          ('vhd', ('VHD - Virtual Hard Disk')),
#          ('vmdk', _('VMDK - Virtual Machine Disk')),
#     ]
#}

# The IMAGE_CUSTOM_PROPERTY_TITLES settings is used to customize the titles for
# image custom property attributes that appear on image detail pages.
IMAGE_CUSTOM_PROPERTY_TITLES = {
    "architecture": _("Architecture"),
    "kernel_id": _("Kernel ID"),
    "ramdisk_id": _("Ramdisk ID"),
    "image_state": _("Euca2ools state"),
    "project_id": _("Project ID"),
    "image_type": _("Image Type"),
}
```

```
# The IMAGE_RESERVED_CUSTOM_PROPERTIES setting is used to specify which image
# custom properties should not be displayed in the Image Custom Properties
# table.
IMAGE_RESERVED_CUSTOM_PROPERTIES = []

# OPENSTACK_ENDPOINT_TYPE specifies the endpoint type to use for the endpoints
# in the Keystone service catalog. Use this setting when Horizon is running
# external to the OpenStack environment. The default is 'publicURL'.
#OPENSTACK_ENDPOINT_TYPE = "publicURL"

# SECONDARY_ENDPOINT_TYPE specifies the fallback endpoint type to use in the
# case that OPENSTACK_ENDPOINT_TYPE is not present in the endpoints
# in the Keystone service catalog. Use this setting when Horizon is running
# external to the OpenStack environment. The default is None. This
# value should differ from OPENSTACK_ENDPOINT_TYPE if used.
#SECONDARY_ENDPOINT_TYPE = "publicURL"

# The number of objects (Swift containers/objects or images) to display
# on a single page before providing a paging element (a "more" link)
# to paginate results.
API_RESULT_LIMIT = 1000
API_RESULT_PAGE_SIZE = 20

# The size of chunk in bytes for downloading objects from Swift
SWIFT_FILE_TRANSFER_CHUNK_SIZE = 512 * 1024

# Specify a maximum number of items to display in a dropdown.
DROPDOWN_MAX_ITEMS = 30

# The timezone of the server. This should correspond with the timezone
# of your entire OpenStack installation, and hopefully be in UTC.
TIME_ZONE = "Asia/Shanghai"

# When launching an instance, the menu of available flavors is
# sorted by RAM usage, ascending. If you would like a different sort order,
# you can provide another flavor attribute as sorting key. Alternatively, you
# can provide a custom callback method to use for sorting. You can also provide
# a flag for reverse sort. For more info, see
# http://docs.python.org/2/library/functions.html#sorted
#CREATE_INSTANCE_FLAVOR_SORT = {
#     'key': 'name',
#      # or
#     'key': my_awesome_callback_method,
#     'reverse': False,
#}

# Set this to True to display an 'Admin Password' field on the Change Password
```

```
# form to verify that it is indeed the admin logged-in who wants to change
# the password.
#ENFORCE_PASSWORD_CHECK = False

# Modules that provide /auth routes that can be used to handle different types
# of user authentication. Add auth plugins that require extra route handling to
# this list.
#AUTHENTICATION_URLS = [
#     'openstack_auth.urls',
#]

# The Horizon Policy Enforcement engine uses these values to load per service
# policy rule files. The content of these files should match the files the
# OpenStack services are using to determine role based access control in the
# target installation.


# Map of local copy of service policy files.
# Please insure that your identity policy file matches the one being used on
# your keystone servers. There is an alternate policy file that may be used
# in the Keystone v3 multi-domain case, policy.v3cloudsample.json.
# This file is not included in the Horizon repository by default but can be
# found at
# http://git.openstack.org/cgit/openstack/keystone/tree/etc/ \
# policy.v3cloudsample.json
# Having matching policy files on the Horizon and Keystone servers is essential
# for normal operation. This holds true for all services and their policy files.
POLICY_FILES_PATH = '/etc/openstack-dashboard'
POLICY_FILES_PATH = '/etc/openstack-dashboard'
# Map of local copy of service policy files
#POLICY_FILES = {
#     'identity': 'keystone_policy.json',
#     'compute': 'nova_policy.json',
#     'volume': 'cinder_policy.json',
#     'image': 'glance_policy.json',
#     'orchestration': 'heat_policy.json',
#     'network': 'neutron_policy.json',
#     'telemetry': 'ceilometer_policy.json',
#}

# Trove user and database extension support. By default support for
# creating users and databases on database instances is turned on.
# To disable these extensions set the permission here to something
# unusable such as ["!"].
#TROVE_ADD_USER_PERMS = []
#TROVE_ADD_DATABASE_PERMS = []
```

```
# Change this patch to the appropriate static directory containing
# two files: _variables.scss and _styles.scss
#CUSTOM_THEME_PATH = 'themes/default'

LOGGING = {
    'version': 1,
    # When set to True this will disable all logging except
    # for loggers specified in this configuration dictionary. Note that
    # if nothing is specified here and disable_existing_loggers is True,
    # django.db.backends will still log unless it is disabled explicitly.
    'disable_existing_loggers': False,
    'handlers': {
        'null': {
            'level': 'DEBUG',
            'class': 'django.utils.log.NullHandler',
        },
        'console': {
            # Set the level to "DEBUG" for verbose output logging.
            'level': 'INFO',
            'class': 'logging.StreamHandler',
        },
    },
    'loggers': {
        # Logging from django.db.backends is VERY verbose, send to null
        # by default.
        'django.db.backends': {
            'handlers': ['null'],
            'propagate': False,
        },
        'requests': {
            'handlers': ['null'],
            'propagate': False,
        },
        'horizon': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'openstack_dashboard': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'novaclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
```

```
    },
    'cinderclient': {
        'handlers': ['console'],
        'level': 'DEBUG',
        'propagate': False,
    },
    'keystoneclient': {
        'handlers': ['console'],
        'level': 'DEBUG',
        'propagate': False,
    },
    'glanceclient': {
        'handlers': ['console'],
        'level': 'DEBUG',
        'propagate': False,
    },
    'neutronclient': {
        'handlers': ['console'],
        'level': 'DEBUG',
        'propagate': False,
    },
    'heatclient': {
        'handlers': ['console'],
        'level': 'DEBUG',
        'propagate': False,
    },
    'ceilometerclient': {
        'handlers': ['console'],
        'level': 'DEBUG',
        'propagate': False,
    },
    'troveclient': {
        'handlers': ['console'],
        'level': 'DEBUG',
        'propagate': False,
    },
    'swiftclient': {
        'handlers': ['console'],
        'level': 'DEBUG',
        'propagate': False,
    },
    'openstack_auth': {
        'handlers': ['console'],
        'level': 'DEBUG',
        'propagate': False,
    },
    'nose.plugins.manager': {
```

```
                'handlers': ['console'],
                'level': 'DEBUG',
                'propagate': False,
            },
            'django': {
                'handlers': ['console'],
                'level': 'DEBUG',
                'propagate': False,
            },
            'iso8601': {
                'handlers': ['null'],
                'propagate': False,
            },
            'scss': {
                'handlers': ['null'],
                'propagate': False,
            },
        }
}

# 'direction' should not be specified for all_tcp/udp/icmp.
# It is specified in the form.
SECURITY_GROUP_RULES = {
    'all_tcp': {
        'name': _('All TCP'),
        'ip_protocol': 'tcp',
        'from_port': '1',
        'to_port': '65535',
    },
    'all_udp': {
        'name': _('All UDP'),
        'ip_protocol': 'udp',
        'from_port': '1',
        'to_port': '65535',
    },
    'all_icmp': {
        'name': _('All ICMP'),
        'ip_protocol': 'icmp',
        'from_port': '-1',
        'to_port': '-1',
    },
    'ssh': {
        'name': 'SSH',
        'ip_protocol': 'tcp',
        'from_port': '22',
        'to_port': '22',
    },
```

```
'smtp': {
    'name': 'SMTP',
    'ip_protocol': 'tcp',
    'from_port': '25',
    'to_port': '25',
},
'dns': {
    'name': 'DNS',
    'ip_protocol': 'tcp',
    'from_port': '53',
    'to_port': '53',
},
'http': {
    'name': 'HTTP',
    'ip_protocol': 'tcp',
    'from_port': '80',
    'to_port': '80',
},
'pop3': {
    'name': 'POP3',
    'ip_protocol': 'tcp',
    'from_port': '110',
    'to_port': '110',
},
'imap': {
    'name': 'IMAP',
    'ip_protocol': 'tcp',
    'from_port': '143',
    'to_port': '143',
},
'ldap': {
    'name': 'LDAP',
    'ip_protocol': 'tcp',
    'from_port': '389',
    'to_port': '389',
},
'https': {
    'name': 'HTTPS',
    'ip_protocol': 'tcp',
    'from_port': '443',
    'to_port': '443',
},
'smtps': {
    'name': 'SMTPS',
    'ip_protocol': 'tcp',
    'from_port': '465',
    'to_port': '465',
```

```
    },
    'imaps': {
        'name': 'IMAPS',
        'ip_protocol': 'tcp',
        'from_port': '993',
        'to_port': '993',
    },
    'pop3s': {
        'name': 'POP3S',
        'ip_protocol': 'tcp',
        'from_port': '995',
        'to_port': '995',
    },
    'ms_sql': {
        'name': 'MS SQL',
        'ip_protocol': 'tcp',
        'from_port': '1433',
        'to_port': '1433',
    },
    'mysql': {
        'name': 'MYSQL',
        'ip_protocol': 'tcp',
        'from_port': '3306',
        'to_port': '3306',
    },
    'rdp': {
        'name': 'RDP',
        'ip_protocol': 'tcp',
        'from_port': '3389',
        'to_port': '3389',
    },
}

# Deprecation Notice:
#
# The setting FLAVOR_EXTRA_KEYS has been deprecated.
# Please load extra spec metadata into the Glance Metadata Definition Catalog.
#
# The sample quota definitions can be found in:
# <glance_source>/etc/metadefs/compute-quota.json
#
# The metadata definition catalog supports CLI and API:
#   $glance --os-image-api-version 2 help md-namespace-import
#   $glance-manage db_load_metadefs <directory_with_definition_files>
#
```

```
# See Metadata Definitions on: http://docs.openstack.org/developer/glance/

# Indicate to the Sahara data processing service whether or not
# automatic floating IP allocation is in effect. If it is not
# in effect, the user will be prompted to choose a floating IP
# pool for use in their cluster. False by default. You would want
# to set this to True if you were running Nova Networking with
# auto_assign_floating_ip = True.
#SAHARA_AUTO_IP_ALLOCATION_ENABLED = False

# The hash algorithm to use for authentication tokens. This must
# match the hash algorithm that the identity server and the
# auth_token middleware are using. Allowed values are the
# algorithms supported by Python's hashlib library.
#OPENSTACK_TOKEN_HASH_ALGORITHM = 'md5'

# Hashing tokens from Keystone keeps the Horizon session data smaller, but it
# doesn't work in some cases when using PKI tokens. Uncomment this value and
# set it to False if using PKI tokens and there are 401 errors due to token
# hashing.
#OPENSTACK_TOKEN_HASH_ENABLED = True

# AngularJS requires some settings to be made available to
# the client side. Some settings are required by in-tree / built-in horizon
# features. These settings must be added to REST_API_REQUIRED_SETTINGS in the
# form of ['SETTING_1','SETTING_2'], etc.
#
# You may remove settings from this list for security purposes, but do so at
# the risk of breaking a built-in horizon feature. These settings are required
# for horizon to function properly. Only remove them if you know what you
# are doing. These settings may in the future be moved to be defined within
# the enabled panel configuration.
# You should not add settings to this list for out of tree extensions.
# See: https://wiki.openstack.org/wiki/Horizon/RESTAPI
REST_API_REQUIRED_SETTINGS = ['OPENSTACK_HYPERVISOR_FEATURES']

# Additional settings can be made available to the client side for
# extensibility by specifying them in REST_API_ADDITIONAL_SETTINGS
# !! Please use extreme caution as the settings are transferred via HTTP/S
# and are not encrypted on the browser. This is an experimental API and
# may be deprecated in the future without notice.
#REST_API_ADDITIONAL_SETTINGS = []
# DISALLOW_IFRAME_EMBED can be used to prevent Horizon from being embedded
# within an iframe. Legacy browsers are still vulnerable to a Cross-Frame
# Scripting (XFS) vulnerability, so this option allows extra security hardening
```

```
# where iframes are not used in deployment. Default setting is True.
# For more information see:
# http://tinyurl.com/anticlickjack
#DISALLOW_IFRAME_EMBED = True
EOF
#重启 httpd 服务
systemctl restart httpd
```

25.16　Openstack GUI 配置

通过浏览器访问 http://192.168.1.120/dashboard/，如图 25-33 所示，输入用户密码 demo 或 admin 即可登录访问。



图 25-33　Openstack 管理平台登录界面

登录 Openstack Web 平台，查看所有节点的概况，如图 25-34 所示。



(a) Openstack Web节点运行概况(1)

图 25-34　Openstack Web 节点运行概况



(b) Openstack Web节点运行概况(2)

(c) Openstack Web节点运行概况(3)

(d) Openstack Web节点运行概况(4)

图 25-34 （续）

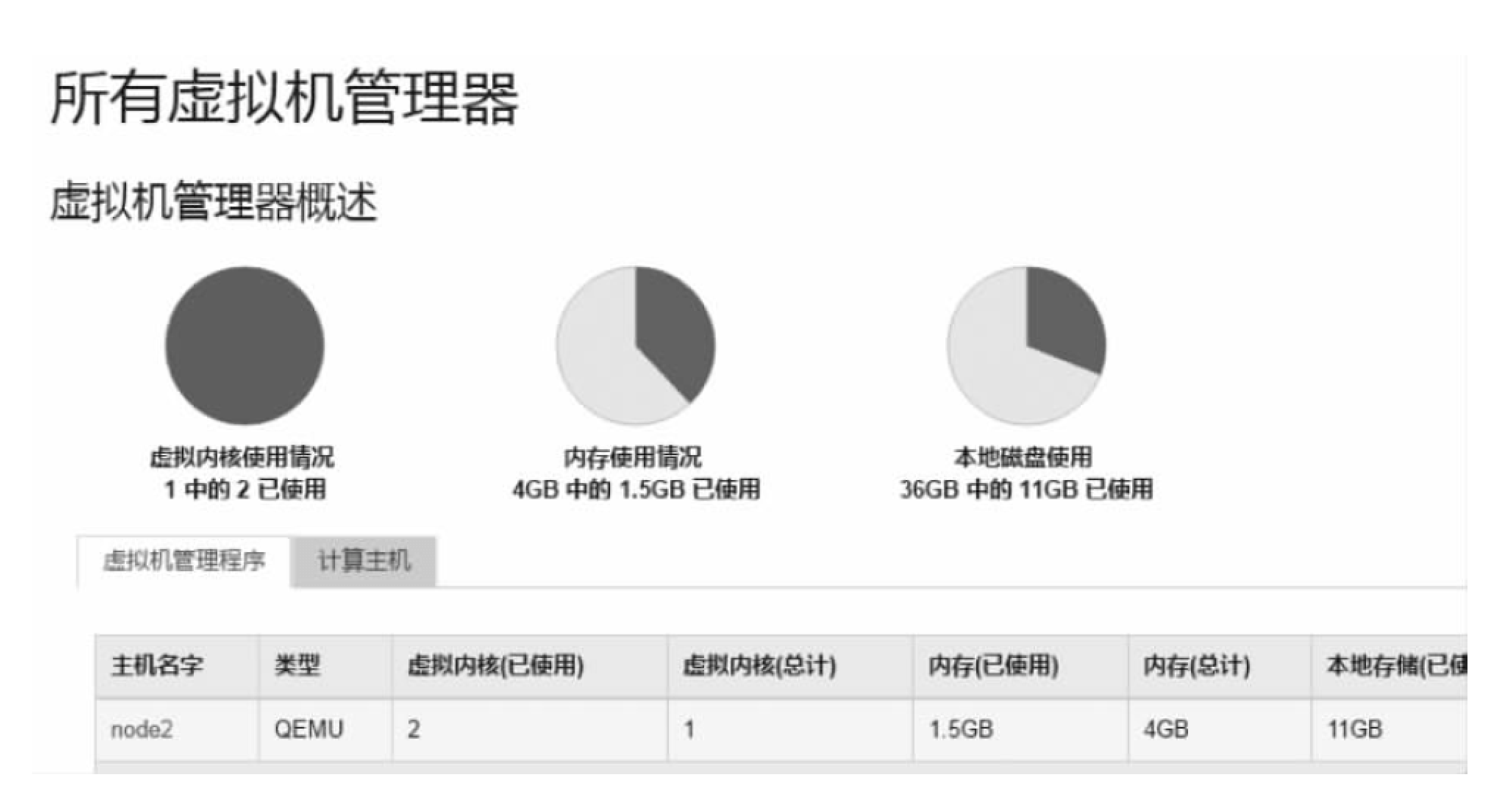
查看 Openstack 云主机类型，默认有 5 种虚拟机创建类型，可以根据实际情况修改或创建虚拟机的类型，如图 25-35 所示。

查看所有虚拟机占用的资源及每个节点资源情况，如图 25-36 所示。



	云主机类型名称	虚拟内核	内存	根磁盘	临时磁盘	交换盘空间	ID	公有	元数据
	m1.tiny	1	512MB	1GB	0GB	0 MB	1	True	{}
	m1.small	1	512MB	10GB	0GB	0 MB	c13fce6e-1210-4395-b8a9-a4b0e415c7a3	True	{}
	m1.medium	2	4GB	40GB	0GB	0 MB	3	True	{}
	m1.large	4	8GB	80GB	0GB	0 MB	4	True	{}
	m1.xlarge	8	16GB	160GB	0GB	0 MB	5	True	{}

图 25-35 Openstack 虚拟机类型



主机名字	类型	虚拟内核(已使用)	虚拟内核(总计)	内存(已使用)	内存(总计)	本地存储(已使
node2	QEMU	2	1	1.5GB	4GB	11GB

图 25-36 Openstack 所有虚拟机资源情况

可以根据企业需要，自定义创建多台虚拟机，创建方法如图 25-37 所示。



(a) Openstack创建虚拟机

(b) Openstack创建虚拟机列表

(c) Openstack创建虚拟机运行列表(1)

图 25-37　Openstack 创建虚拟机方法



(d) Openstack创建虚拟机运行列表(2)

(e) Openstack虚拟机登录界面

图 25-37 （续）

25.17　Openstack 核心流程

Openstack 由不同的组件组成，通过以上学习，相信很多读者对于学习 Openstack 这门技术来讲，最难之处莫过于对各个组件的关系和功能的理解，以及整个 Openstack 创建虚拟机流程。如图 25-38 所示为 Openstack 创建一台虚拟主机的完整流程。

如图 25-38 所示，Openstack 私有云创建一台完整虚拟机，详解如下：

- 使用者或客户端：Web 页面或 Horizon、命令行 nova 指令。
- Nova API：用于接收和处理客户端发送的 HTTP 请求。
- Nova scheduler：Nova 调度宿主机的服务，决定虚拟机创建的各节点。
- Nova compute：Nova 核心的服务，负责虚拟机的生命周期的管理。
- Nova conductor：数据访问权限的控制操作，可以理解为数据库代理服务。
- Nova cert：管理证书，为了兼容 aws。

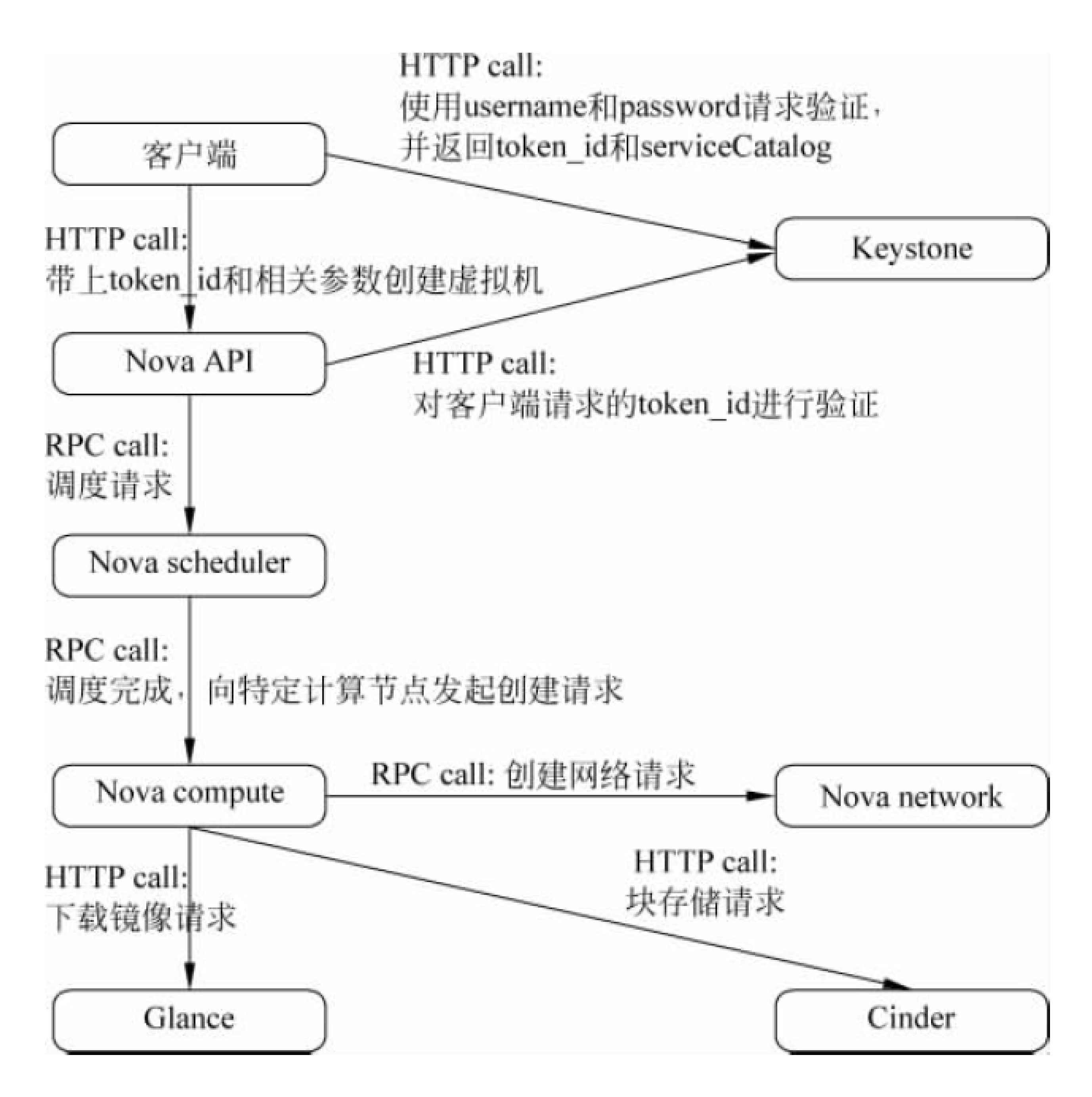


图 25-38　Openstack 虚拟机创建流程

- Nova vncproxy 和 consoleauth：主要提供 VNC 及 auth 认证控制台。

Openstack 组件不同的模块之间是通过 HTTP 请求 rest API 服务，同一模块不同组件之间通过 RPC 远程调用，通过 RabbitMQ 消息异步通信来实现。

如图 25-39 所示为 Openstack 所有组件架构图及调用步骤，详解如下：

(1) 客户端基于用户名和密码请求认证。

(2) Keystone 查询 Keystone 数据库 user 表中保存的 user 相关信息，包括 password 加密后的 hash 值，并返回一个 token_id(令牌)和 serviceCatalog。

(3) 客户端带上 Keystone 返回的 token_id 和创建虚机的相关参数，post 请求 Nova API 创建虚拟机。

(4) Nova API 接收到请求后，使用请求携带的 token_id 来访问该 API，以验证请求是否有效。

(5) Keystone 验证通过后返回更新后的认证信息。

7 Nova API 检查创建虚拟机参数是否有效与合法，检查虚拟机 name 是否符合命名规范，flavor_id 是否在数据库中存在，image_uuid 是否是正确的 uuid 格式，检查 instance、vcpu、ram 的数量是否超过配额。

(7) 当所有传参都有效合法时，更新 Nova 数据库，新建一条 instance 记录，vm_states 设为 BUILDING，task_state 设为 SCHEDULING。

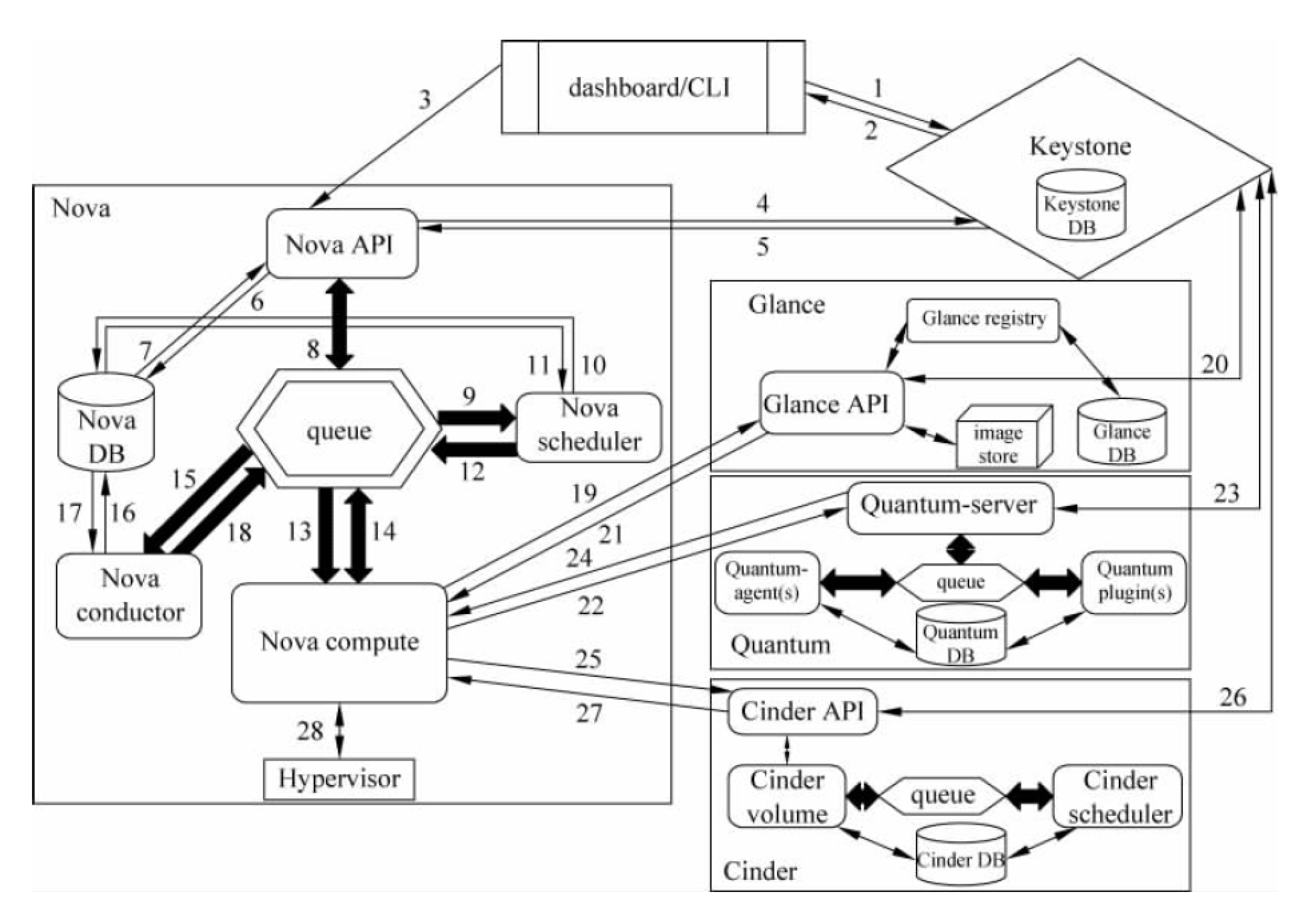


图 25-39 Openstack 所有组件调用步骤

(8) Nova API 远程调用传递请求，将参数给 Nova scheduler，把消息“请给我创建一台虚拟机”丢到 MQ 消息队列，然后定期查询虚机的状态。

(9) Nova scheduler 从 MQ 消息队列中获取到这条消息。

(10) Nova scheduler 访问 Nova 数据库，通过调度算法，过滤出一些合适的计算节点，然后进行排序。

(11) 更新虚拟机节点信息，返回一个最优节点 ID 给 Nova scheduler。

(12) Nova scheduler 选定 host 之后，通过 rpc 调用 Nova compute 服务，把“创建虚机请求”消息发到 MQ 消息队列。

(13) Nova compute 收到创建虚拟机请求的消息，通过定时任务，定期从数据库中查找到运行在该节点上的所有虚拟机信息，统计得到空闲内存大小和空闲磁盘大小，然后更新数据库 compute_node 信息，以保证调度的准确性。

(14) Nova compute 通过 rpc 查询 Nova 数据库中虚拟机的信息，如主机模板和 ID。

(15) Nova conductor 从 MQ 消息队列中拿到请求查询数据库。

(16) Nova conductor 查询 Nova 数据库。

(17) 数据库返回虚机信息。

(18) Nova compute 从 MQ 消息队列中获取信息。

(19) Nova compute 请求 Glance 的 rest API，下载所需要的镜像，一般是 qcow2，可以自定义创建镜像。

（20）Glance API 会验证请求的 token 的有效性。

（21）Glance API 返回镜像信息给 Nova compute。

（22）Nova compute 请求 Neutron API 配置网络，获取虚机 IP 地址等信息。

（23）验证 token 的有效性。

（24）Neutron 返回网络信息。

（25）～（27）步骤同 Glance、Neutron 验证 token 返回块设备信息。

（28）根据获取的虚拟机信息，生成 KVM 所需的.xml 文件，写入 libvirt.xml 文件，然后调用 libvirt driver 去使用 libvirt.xml 文件启动虚拟机。